Statistical Science
2010, Vol. 25, No. 1, 88–106
DOI: 10.1214/10-STS325

Particle Learning and Smoothing

Carlos M. Carvalho, Michael S. Johannes, Hedibert F. Lopes and Nicholas G. Polson

Abstract. Particle learning (PL) provides state filtering, sequential parameter learning and smoothing in a general class of state space models. Our approach extends existing particle methods by incorporating the estimation of static parameters via a fully-adapted filter that utilizes conditional sufficient statistics for parameters and/or states as particles. State smoothing in the presence of parameter uncertainty is also solved as a by-product of PL. In a number of examples, we show that PL outperforms existing particle filtering alternatives and proves to be a competitor to MCMC.

Key words and phrases: Mixture Kalman filter, parameter learning, particle learning, sequential inference, smoothing, state filtering, state space models.

1. INTRODUCTION

There are two statistical inference problems associated with state space models. The first is sequential state filtering and parameter learning, which is characterized by the joint posterior distribution of parameters and states at each point in time. The second is state smoothing, which is characterized by the distribution of the states, conditional on all available data, marginalizing out the unknown parameters.

In linear Gaussian models, assuming knowledge about the system parameters, the Kalman filter (Kalman, 1960) provides the standard analytical recursions for filtering and smoothing (West and Harrison, 1997). For more general model specifications, conditional on parameters, it is common to use sequential Monte Carlo methods known as particle filters to approximate the sequence of filtering distributions (see Doucet, de Freitas and Gordon, 2001 and Cappé, Godsill and Moulines, 2007). As for smoothing, the posterior for states is typically approximated via Markov chain Monte Carlo (MCMC) methods as developed by Carlin, Polson and Stoffer (1992), Carter and Kohn (1994) and Frühwirth-Schnatter (1994).

In this paper we propose a new approach, called particle learning (PL), for approximating the sequence of filtering and smoothing distributions in light of parameter uncertainty for a wide class of state space models. The central idea behind PL is the creation of a particle algorithm that directly samples from the particle approximation to the joint posterior distribution of states and conditional sufficient statistics for fixed parameters in a fully-adapted resample–propagate framework.

In terms of models, we consider Gaussian Dynamic Linear Models (DLMs) and conditionally Gaussian (CDLMs). In these class of models, PL is de-

Carlos M. Carvalho is Assistant Professor of Econometrics and Statistics, University of Chicago Booth School of Business, 5807 South Woodlawn Avenue, Chicago, Illinois 60637, USA e-mail: carlos.carvalho@chicagobooth.edu. Michael Johannes is Roger F. Murray Associate Professor of Finance, Graduate School of Business, Columbia University, 3022 Broadway, Uris Hall 424, New York, NY 10027, USA e-mail: mj335@columbia.edu. Hedibert F. Lopes is Associate Professor of Econometrics and Statistics, University of Chicago Booth School of Business, 5807 South Woodlawn Avenue Chicago, Illinois 60637, USA e-mail: hlopes@chicagobooth.edu. Nicholas G. Polson is Professor of Econometrics and Statistics, University of Chicago Booth School of Business, 5807 South Woodlawn Avenue Chicago, Illinois 60637, USA e-mail: ngp@chicagobooth.edu.

fined over both state and parameter sufficient statistics. This is a generalization of the mixture Kalman filter (MKF) of Chen and Liu (2000) that allows for parameter learning. Additionally, we show that PL can handle nonlinearities in the state evolutions, dramatically widening the class of models that MKF particle methods apply to. Finally, we extend the smoothing results of Godsill, Doucet and West (2004) to sequential parameter learning and to all the models considered.

In a series of simulation studies, we provide significant empirical evidence that PL dominates the standard particle filtering alternatives in terms of estimation accuracy and that it can be seen as a true competitor to MCMC strategies.

The paper starts in Section 2, with a brief review of the most popular particle filters that represent the building blocks for the development of PL in Section 3. Section 4 in entirely dedicated to the application of PL to CDLMs followed by possible extensions to nonlinear alternatives in Section 5. Section 6 presents a series of experiments benchmarking the performance of PL and highlighting its advantages over currently used alternatives.

2. PARTICLE FILTERING IN STATE SPACE MODELS

Consider a general state space model defined by the observation and evolution equations:

$$y_{t+1} \sim p(y_{t+1}|x_{t+1}, \theta),$$
$$x_{t+1} \sim p(x_{t+1}|x_t, \theta),$$

with initial state distribution $p(x_0|\theta)$ and prior $p(\theta)$. In the above notation, states at time t are represented by x_t while the static parameters are denoted by θ. The sequential state filtering and parameter learning problem is solved by the sequence of joint posterior distributions, $p(x_t, \theta|y^t)$, where $y^t = (y_1, \ldots, y_t)$ is the set of observations up to time t.

Particle methods use a discrete representation of $p(x_t, \theta|y^t)$ via

$$p^N(x_t, \theta|y^t) = \frac{1}{N}\sum_{i=1}^{N} \delta_{(x_t,\theta)^{(i)}},$$

where $(x_t, \theta)^{(i)}$ is the state and parameter particle vector and $\delta_{(.)}$ is the Dirac measure, representing the distribution degenerate at the N particles. Given this approximation, the key problem is how to sample from this joint distribution sequentially as new data arrives. This step is complicated because the state's propagation depends on the parameters, and vice versa. To circumvent the codependence in a joint draw, it is common to use proposal distributions in a sequence of importance sampling steps. We now review the main approaches of this general sequential Monte Carlo strategy first for pure filtering and then with parameter learning.

2.1 Pure Filtering Review

We start by considering the pure filtering problem, where it is assumed that the set of parameters θ is known. Although less relevant in many areas of application, this is the traditional engineering application where both the Kalman filter and original particle filters were developed.

The bootstrap filter In what can be considered the seminal work in the particle filtering literature, Gordon, Salmond and Smith (1993) developed a strategy based on a sequence of importance sampling steps where the proposal is defined by the prior for the states. This algorithm uses the following representation of the filtering density:

$$p(x_{t+1}|y^{t+1}) \propto p(y_{t+1}|x_{t+1})p(x_{t+1}|y^t),$$

where the state predictive is

$$p(x_{t+1}|y^t) = \int p(x_{t+1}|x_t)p(x_t|y^t)\,dx_t.$$

Starting with a particle approximation of $p(x_t|y^t)$, draws from $p(x_{t+1}|y^t)$ are obtained by propagating the particles forward via the evolution equation $p(x_{t+1}|x_t)$, leading to importance sampling weights that are proportional to like likelihood $p(y_{t+1}|x_{t+1})$. The bootstrap filter can be summarized by the following:

Bootstrap filter (BF).

Step 1 (Propagate). $\{x_t^{(i)}\}_{i=1}^N$ to $\{\tilde{x}_{t+1}^{(i)}\}_{i=1}^N$ via $p(x_{t+1}|x_t)$.

Step 2 (Resample). $\{x_{t+1}^{(i)}\}_{i=1}^N$ from $\{\tilde{x}_{t+1}^{(i)}\}_{i=1}^N$ with weights $w_{t+1}^{(i)} \propto p(y_{t+1}|\tilde{x}_{t+1}^{(i)})$.

Resampling in the second stage is an optional step, as any quantity of interest could be computed more accurately by the use of the particles and its associated weights. Resampling has been used as a way to avoid the decay in the particle approximation and we refer the reader to Liu and Chen (1998) for a careful discussion of its merits. Throughout our

work we describe all filters with a resampling step, as this is the central idea to our particle learning strategy introduced below. Notice, therefore, that we call BF a *propagate–resample* filter due to the order of operation of its steps:

AUXILIARY PARTICLE FILTER (APF).

Step 1 (Resample). $\{\tilde{x}_t^{(i)}\}_{i=1}^N$ from $\{x_t^{(i)}\}_{i=1}^N$ with weights

$$\tilde{w}_{t+1}^{(i)} \propto p(y_{t+1}|g(x_t^{(i)})).$$

Step 2 (Propagate). $\{\tilde{x}_t^{(i)}\}_{i=1}^N$ to $\{\tilde{x}_{t+1}^{(i)}\}_{i=1}^N$ via $p(x_{t+1}|\tilde{x}_t)$.

Step 3 (Resample). $\{\tilde{x}_{t+1}^{(i)}\}_{i=1}^N$ with weights

$$w_{t+1}^{(i)} \propto \frac{p(y_{t+1}|\tilde{x}_{t+1}^{(i)})}{p(y_{t+1}|g(\tilde{x}_t^{(i)}))}.$$

Auxiliary particle filter (APF) The APF of Pitt and Shephard (1999) uses a different representation of the joint filtering distribution of (x_t, x_{t+1}) as

$$\begin{aligned} &p(x_t, x_{t+1}|y^{t+1}) \\ &\quad \propto p(x_{t+1}|x_t, y^{t+1})p(x_t|y^{t+1}) \\ &\quad = p(x_{t+1}|x_t, y^{t+1})p(y_{t+1}|x_t)p(x_t|y^t). \end{aligned}$$

Our view of the APF is as follows: starting with a particle approximation of $p(x_t|y^t)$, draws from the smoothed distribution of $p(x_t|y^{t+1})$ are obtained by resampling the particles with weights proportional to the predictive $p(y_{t+1}|x_t)$. These resampled particles are then propagated forward via $p(x_{t+1}|x_t, y^{t+1})$. The APF is therefore a *resample–propagate* filter. Using the terminology of Pitt and Shephard (1999), the above representation is an optimal, *fully adapted* strategy where exact samples from $p^N(x_{t+1}|y^{t+1})$ were obtained, avoiding an importance sampling step. This is possible if both the predictive and propagation densities were available for evaluation and sampling.

In general, this is not the case and Pitt and Shephard proposed the use of an importance function $p(y_{t+1}|\hat{\mu}_{t+1} = g(x_t))$ for the resampling step based on a *best guess* for x_{t+1} defined by $\hat{\mu}_{t+1} = g(x_t)$. This could be, for example, the expected value, the median or mode of the state evolution. The resampled particles would then be propagated with a second proposal defined by $p(x_{t+1}|x_t)$, leading to the following algorithm:

Two main ideas make the APF an attractive approach: (i) the current observation y_{t+1} is used in the proposal of the first resampling step and (ii) due to the pre-selection in step 1, only "good" particles are propagated forward. The importance of this second point will prove very relevant in the success of our proposed approach.

2.2 Sequential Parameter Learning Review

Sequential estimation of fixed parameters θ is notoriously difficult. Simply including θ in the particle set is a natural but unsuccessful solution, as the absence of a state evolution implies that we will be left with an ever-decreasing set of atoms in the particle approximation for $p(\theta|y^t)$. Important developments in this direction appear in Liu and West (2001), Storvik (2002), Fearnhead (2002), Polson, Stroud and Müller (2008), Johannes and Polson (2008) and Johannes, Polson and Yae (2008), to cite a few. We now review two popular alternatives to learn about θ:

STORVIK'S FILTER.

Step 1 (Propagate). $\{x_t^{(i)}\}_{i=1}^N$ to $\{\tilde{x}_{t+1}^{(i)}\}_{i=1}^N$ via $q(x_{t+1}|x_t^{(i)}, \theta^{(i)}, y^{t+1})$.

Step 2 (Resample). $\{(x_{t+1}, s_t)^{(i)}\}_{i=1}^N$ from $\{(\tilde{x}_{t+1}, s_t)^{(i)}\}_{i=1}^N$ with weights

$$w_{t+1}^{(i)} \propto \frac{p(y_{t+1}|\tilde{x}_{t+1}^{(i)}, \theta)p(\tilde{x}_{t+1}^{(i)}|x_t^{(i)}, \theta)}{q(\tilde{x}_{t+1}^{(i)}|x_t^{(i)}, \theta, y^{t+1})}.$$

Step 3 (Propagate). Sufficient statistics $s_{t+1}^{(i)} = \mathcal{S}(s_t^{(i)}, x_{t+1}^{(i)}, y_{t+1})$.

Step 4 (Sample). $\theta^{(i)}$ from $p(\theta|s_{t+1}^{(i)})$.

Storvik's filter Storvik (2002) (similar ideas appear in Fearnhead, 2002) assumes that the posterior distribution of θ given x^t and y^t depends on a low-dimensional set of sufficient statistics that can be recursively updated. This recursion for sufficient statistics is defined by $s_{t+1} = \mathcal{S}(s_t, x_{t+1}, y_{t+1})$, leading to the above algorithm. Notice that the proposal $q(\cdot)$ is conditional on y_{t+1}, but this is still a propagate–resample filter.

Liu and West's filter Liu and West (2001) suggest a kernel approximation $p(\theta|y^t)$ based on a mixture of multivariate normals. This idea is used in the context of the APF. Specifically, let $\{(x_t, \theta_t)^{(i)}\}_{i=1}^N$ be particle draws from $p(x_t, \theta|y^t)$. Hence, the posterior for θ can be approximated by the mixture distribution

$$p(\theta|y^t) = \sum_{j=1}^N N(m^{(j)}; h^2 V_t),$$

where $m^{(j)} = a\theta_t^{(j)} + (1-a)\tilde{\theta}_t$, $\tilde{\theta}_t = \sum_{j=1}^N \theta_t^{(j)}/N$ and $V_t = \sum_{j=1}^N (\theta_t^{(j)} - \bar{\theta}_t)(\theta_t^{(j)} - \bar{\theta}_t)'/N$. The constants a and h measure, respectively, the extent of the shrinkage and the degree of overdispersion of the mixture (see Liu and West, 2001 for a detailed discussion of the choice of a and h). The idea is to use the mixture approximation to generate fresh samples from the current posterior in an attempt to avoid particle decay. The algorithm is summarized in the next page. The main attraction of Liu and West's filter is its generality, as it can be implemented in any state-space model. It also takes advantage of APF's resample–propagate framework and can be considered a benchmark in the current literature:

Liu and West's filter.

Step 1 (Resample). $\{(\tilde{x}_t, \tilde{\theta}_t)^{(i)}\}_{i=1}^N$ from $\{(x_t, \theta_t)^{(i)}\}_{i=1}^N$ with weights

$$w_{t+1}^{(i)} \propto p(y_{t+1}|g(x_t^{(i)}), m^{(i)}).$$

Step 2 (Propagate).

(2.1) $\{\tilde{\theta}_t^{(i)}\}_{i=1}^N$ to $\{\hat{\theta}_{t+1}^{(i)}\}_{i=1}^N$ via $N(\tilde{m}^{(i)}, V)$;
(2.2) $\{\tilde{x}_t^{(i)}\}_{i=1}^N$ to $\{\hat{x}_{t+1}^{(i)}\}_{i=1}^N$ via $p(x_{t+1}|\tilde{x}_t^{(i)}, \hat{\theta}_{t+1}^{(i)})$.

Step 3 (Resample). $\{(x_{t+1}, \theta_{t+1})^{(i)}\}_{i=1}^N$ from $\{(\hat{x}_{t+1}, \hat{\theta}_{t+1})^{(i)}\}_{i=1}^N$ with weights

$$w_{t+1}^{(i)} \propto \frac{p(y_{t+1}|\hat{x}_{t+1}^{(i)}, \hat{\theta}_{t+1}^{(i)})}{p(y_{t+1}|g(\tilde{x}_t^{(i)}), \tilde{m}^{(i)})}.$$

3. PARTICLE LEARNING AND SMOOTHING

Our proposed approach for filtering and learning relies on two main insights: (i) conditional sufficient statistics are used to represent the posterior of θ. Whenever possible, sufficient statistics for the latent states are also introduced, increasing the efficiency of our algorithm by reducing the variance of sampling weights in what can be called a Rao–Blackwellized filter. (ii) We use a resample–propagate framework and attempt to build perfectly adapted filters whenever possible in trying to obtain exact samples from our particle approximation when moving from $p^N(x_t, \theta|y^t)$ to $p^N(x_{t+1}, \theta|y^{t+1})$. This avoids sample importance re-sampling and the associated "decay" in the particle approximation. As with any particle method, there will be accumulation of Monte Carlo error and this has to be analyzed on a case-by-case basis. Simply stated, PL builds on the ideas of Johannes and Polson (2008) and creates a fully adapted extension of the APF to deal with parameter uncertainty. Without delays, PL can be summarized as follows, with details provided in the following sections:

Particle learning.

Step 1 (Resample). $\{\tilde{z}_t^{(i)}\}_{i=1}^N$ from $z_t^{(i)} = (x_t, s_t, \theta)^{(i)}$ with weights $w_t \propto p(y_{t+1}|z_t^{(i)})$.

Step 2 (Propagate). $\tilde{x}_t^{(i)}$ to $x_{t+1}^{(i)}$ via $p(x_{t+1}|\tilde{z}_t^{(i)}, y_{t+1})$.

Step 3 (Propagate). Sufficient statistics $s_{t+1}^{(i)} = \mathcal{S}(\tilde{s}_t^{(i)}, x_{t+1}^{(i)}, y_{t+1})$.

Step 4 (Sample). $\theta^{(i)}$ from $p(\theta|s_{t+1}^{(i)})$.

Due to our initial resampling of states and sufficient statistics, we would end up with a more representative set of propagated sufficient statistics when sampling parameters than Storvik's filter.

3.1 Discussion

Assume that at time t, after observing y^t, we have a particle approximation $p^N(z_t|y^t)$, given by $\{z_t^{(i)}\}_{i=1}^N$. Once y_{t+1} is observed, PL updates the above approximation using the following resample–propagate rule:

$$p(z_t|y^{t+1}) \propto p(y_{t+1}|z_t)p(z_t|y^t) \tag{3.1}$$

and

$$\begin{aligned} p(z_{t+1}|y^{t+1}) = \int & p(s_{t+1}|x_{t+1}, s_t, y_{t+1}) \\ & \cdot p(x_{t+1}|z_t, y_{t+1}) \\ & \cdot p(z_t|y^{t+1})\, dx_{t+1}\, dz_t. \end{aligned} \tag{3.2}$$

From (3.1), we see that an updated approximation $p^N(z_t|y^{t+1})$ can be obtained by resampling the current particles set with weights proportional to the predictive $p(y_{t+1}|z_t)$. This updated approximation is used in (3.2) to generate propagated samples from the posterior $p(x_{t+1}|z_t, y_{t+1})$ that are then used to update s_{t+1}, deterministically, by the recursive map $\mathcal{S}(\cdot)$, which in (3.2) we denote by $p(s_{t+1}|x_{t+1}, s_t, y_{t+1})$. However, since s_t and x_{t+1} are random variables, the conditional sufficient statistics s_{t+1} are also random and are replenished, essentially as a state, in the filtering step. This is the key insight for handling the learning of θ. The particles for s_{t+1} are sequentially updated with resampled s_t particles and propagated and replenished x_{t+1} particles and updated samples

from $p(\theta|s_{t+1})$ can be obtained at the end of the filtering step.

By resampling first we reduce the compounding of approximation errors as the states are propagated after being "informed" by y_{t+1}, as in APF. To clarify the notion of full-adaptation, we can rewrite the problem of updating the particles $\{z_t^{(i)}\}_{i=1}^N$ to $\{z_{t+1}^{(i)}\}_{i=1}^N$ as the problem of obtaining samples from the target $p(x_{t+1}, z_t|y^{t+1})$ based on draws from the proposal $p(z_t|y^{t+1})p(x_{t+1}|z_t, y^{t+1})$, yielding importance weights

$$w_{t+1} \propto \frac{p(x_{t+1}, z_t|y^{t+1})}{p(z_t|y^{t+1})p(x_{t+1}|z_t, y^{t+1})} = 1, \tag{3.3}$$

and therefore, exact draws. Sampling from the proposal is done in two steps: first draws $z_t^{(i)}$ from $p(z_t|y^{t+1})$ are simply obtained by resampling the particles $\{z_t^{(i)}\}_{i=1}^N$ with weights proportional to $p(y_{t+1}|z_t)$; we can then sample $x_{t+1}^{(i)}$ from $p(x_{t+1}|z_t, y^{t+1})$. Finally, updated samples for s_{t+1} are obtained as a function of the samples of x_{t+1}, with weights $1/N$, which prevents particle degeneracies in the estimation of θ. This is a feature of the "resample–propagate" mechanism of PL. Any propagate–resample strategy will lead to decay in the particles of x_{t+1} with significant negative effects on $p^N(\theta|s_{t+1})$. This strategy will only be possible whenever both $p(y_{t+1}|z_t)$ and $p(x_{t+1}|z_t, y^{t+1})$ are analytically tractable, which is the case in the classes of models considered here.

Convergence properties of the algorithm are straightforward to establish. The choice of particle size N to achieve a desired level of accuracy depends, however, on the speed of Monte Carlo accumulation error. In some cases this will be uniformly bounded. In others, a detailed simulation experiment has to be performed. The error will depend on a number of factors. First, the usual signal-to-noise ratio with the smaller the value leads to larger accumulation. Section 4 provides detailed simulation evidence for the models in question. Second, a source of Monte Carlo error can appear from using a particle approximation to the initial state and parameter distribution. This error is common to all particle methods. At its simplest level our algorithm only requires samples $\theta^{(i)}$ from the prior $p(\theta)$. However, a natural class of priors for diffuse situations are mixtures of the form $p(\theta) = \int p(\theta|z_0)p(z_0)\,dz_0$, with the conditional $p(\theta|z_0)$ chosen to be conditionally conjugate. This extra level of analytical tractability can lead to substantial improvements in the initial Monte Carlo error. Particles $z_0^{(i)}$ are drawn from $p(z_0)$ and then resampled from the predictive and then propagated. Mixtures of this form are very flexible and allow for a range of nonconjugate priors. We now turn to specific examples.

EXAMPLE 1 (First order DLM). For illustration, consider first the simple first order dynamic linear model, also known as the local level model (West and Harrison, 1997), where

$$(y_{t+1}|x_{t+1}, \theta) \sim N(x_{t+1}, \sigma^2),$$
$$(x_{t+1}|x_t, \theta) \sim N(x_t, \tau^2),$$

with $\theta = (\sigma^2, \tau^2)$, $x_0 \sim N(m_0, C_0)$, $\sigma^2 \sim \mathrm{IG}(a_0, b_0)$ and $\tau^2 \sim \mathrm{IG}(c_0, d_0)$. The hyperparameters m_0, C_0, a_0, b_0, c_0 and d_0 are kept fixed and known. It is straightforward to show that

$$(y_{t+1}|x_t, \theta) \sim N(x_t, \sigma^2 + \tau^2) \quad \text{and}$$
$$(x_{t+1}|y_{t+1}, x_t, \theta) \sim N(\mu_t, \omega^2),$$

where $\mu_t = \omega^2(\sigma^{-2}y_{t+1} + \tau^{-2}x_t)$, $\omega^{-2} = \sigma^{-2} + \tau^{-2}$. Also, for scales

$$(\sigma^2|y^{t+1}, x^{t+1}) \sim \mathrm{IG}(a_{t+1}, b_{t+1}) \quad \text{and}$$
$$(\tau^2|y^{t+1}, x^{t+1}) \sim \mathrm{IG}(c_{t+1}, d_{t+1}),$$

where $a_{t+1} = a_t + 1/2$, $c_{t+1} = c_t + 1/2$, $b_{t+1} = b_t + 0.5(y_{t+1} - x_{t+1})^2$ and $d_{t+1} = d_t + 0.5(x_{t+1} - x_t)^2$. Therefore, the vector of conditional sufficient statistics s_{t+1} is 5-dimensional and satisfies the following deterministic recursions: $s_{t+1} = s_t + (y_{t+1}^2, y_{t+1}x_{t+1}, x_{t+1}^2, x_t^2, x_{t+1}x_t)$. Finally, notice that, in both, $p(y_{t+1}|x_t)$ and $p(x_{t+1}|x_t, y^{t+1})$ are available for evaluation and sampling, so that a fully adapted version of PL can be implemented.

3.2 State Sufficient Statistics

A more efficient approach, whenever possible, is to marginalize states and just track conditional state sufficient statistics. In the pure filtering case, Chen and Liu (2000) use a similar approach. Here we use the fact that

$$p(x_t|y^t) = \int p(x_t|s_t^x)p(s_t^x|y^t)\,ds_t^x.$$

Thus, we are interested in the distribution $p(s_t^x|y^t)$. The filtering recursions are given by

$$p(s_{t+1}^x|y^{t+1}) = \int p(s_{t+1}^x|s_t^x, x_{t+1}, y_{t+1}) \cdot p(s_t^x, x_{t+1}|y^{t+1})\,ds_t^x\,dx_{t+1}.$$

We can decompose $p(s_t^x, x_{t+1}|y^{t+1})$ as proportional to

$$p(y_{t+1}|s_t^x)p(x_{t+1}|s_t^x, y_{t+1})p(s_t^x|y^t),$$

where we have an extra level of marginalization. Instead of marginalizing x_t, you now marginalize over s_t^x and x_{t+1}. For this to be effective, we need the following conditional posterior:

$$p(x_{t+1}|s_t^x, y_{t+1}) = \int p(x_{t+1}|x_t, y_{t+1})p(x_t|s_t^x)\,dx_t.$$

We can then proceed with the particle learning algorithm. Due to this Rao–Blackwellization step, the weights are flatter in the first stage, that is, $p(y_{t+1}|s_t^x)$ versus $p(y_{t+1}|x_t)$ increasing the efficiency of the algorithm.

EXAMPLE 1 (Cont.). Recalling $(x_t|\theta) \sim N(m_t, C_t)$, then it is straightforward to see that $(y_{t+1}|m_t, C_t, \theta) \sim N(m_t, C_t + \sigma^2 + \tau^2)$, so $s_t^x = (m_t, C_t)$. The recursions for the state sufficient statistics vector s_t^x are the well-known Kalman recursions, that is, $m_{t+1} = (1 - A_{t+1})m_t + A_{t+1}y_{t+1}$ and $C_{t+1} = A_{t+1}\sigma^2$, where $A_{t+1} = (C_t + \tau^2)/(C_t + \tau^2 + \sigma^2)$ is the Kalman gain.

3.3 Smoothing

Smoothing, that is, estimating the states and parameters conditional on all available information, is characterized by $p(x^T, \theta|y^T)$, with T denoting the last observation.

After one sequential pass through the data, our particle approximation computes samples from $p^N(x_t, s_t|y^t)$ for all $t \le T$. However, in many situations, we are required to obtain full smoothing distributions $p(x^T|y^T)$ which are typically carried out by a MCMC scheme. We now show that our filtering strategy provides a direct backward sequential pass to sample from the target smoothing distribution. To compute the marginal smoothing distribution, we write the joint posterior of (x^T, θ) as

$$p(x^T, \theta|y^T) = \prod_{t=1}^{T-1} p(x_t|x_{t+1}, \theta, y^t)p(x_T, \theta|y^T).$$

By Bayes' rule and conditional independence, we have

$$p(x_t|x_{t+1}, \theta, y^t) \propto p(x_{t+1}|x_t, \theta, y^t)p(x_t|\theta, y^t).$$

We can now derive a recursive backward sampling algorithm to jointly sample from $p(x^T, \theta|y^T)$ by sequentially sampling from filtered particles with weights proportional to $p(x_{t+1}|x_t, \theta, y^t)$. In detail, randomly choose, at time T, $(\tilde{x}_T, \tilde{s}_T)$ from the particle approximation $p^N(x_T, s_T|y^T)$ and sample $\tilde{\theta} \sim p(\theta|\tilde{s}_T)$. Then, for $t = T-1, \ldots, 1$, choose $\tilde{x}_t = x_t^{(i)}$ from the filtered particles $\{x_t^{(i)}, i = 1, \ldots, N\}$ with weights $w_{t|t+1}^{(i)} \propto p(\tilde{x}_{t+1}|x_t^{(i)}, \tilde{\theta})$:

PARTICLE SMOOTHING.

Step 1 (Forward filtering). Sample $\{(x^T, \theta)^{(i)}\}_{i=1}^N$ via *particle learning*.

Step 2 (Backwards smoothing). For each pair $(x_T, \theta)^{(i)}$ and $t = T-1, \ldots, 1$, resample $x_t^{(i)}$ from $\{x_t^{(j)}\}_{j=1}^N$ with weights

$$w_{t|t+1}^{(j)} \propto p(x_{t+1}^{(i)}|x_t^{(j)}, \theta^{(i)}).$$

This algorithm is an extension of Godsill, Doucet and West (2004) to state space models where the fixed parameters are unknown. See also Briers, Doucet and Maskell (2010) for an alternative SMC smoother. Both SMC smoothers are $O(TN^2)$, so the computational time to obtain draws from $p(x^T|y^T)$ is expected to be much larger than the computational time to obtain draws from $p(x_t|y^t)$, for $t = 1, \ldots, T$, from standard SMC filters. An $O(TN)$ smoothing algorithm has recently been introduced by Fearnhead, Wyncoll and Tawn (2008).

EXAMPLE 1 (Cont.). For $t = T-1, \ldots, 2, 1$, it is easy to see that $(x_t|x_{t+1}, y^T, \theta) \sim N(a_t, D_t\tau^2)$ and $(x_t|y^T, \theta) \sim N(m_t^T, C_t^T)$, where $a_t = (1 - D_t)m_t + D_t x_{t+1}$ $m_t^T = (1 - D_t)m_t + D_t m_{t+1}^T$, $C_t^T = (1 - D_t)C_t + D_t^2 C_{t+1}^T$, and $D_t = C_t/(C_t + \tau^2)$. Finally, $m_T^T = m_T$ and $C_T^T = C_T$.

3.4 Model Monitoring

The output of PL can be used for sequential predictive problems but is also key in the computation of Bayes factors for model assessment in state space models. Specifically, the marginal predictive for a given model $\mathcal{M}$ can be approximated via

$$p^N(y_{t+1}|y^t, \mathcal{M}) = \frac{1}{N}\sum_{i=1}^N p(y_{t+1}|(x_t, \theta)^{(i)}, \mathcal{M}).$$

This then allows the computation of a SMC approximation to the Bayes factor B_{t+1} or sequential likelihood ratios for competing models $\mathcal{M}_0$ and $\mathcal{M}_1$ (see, e.g., West, 1986):

$$B_{t+1} = \frac{p(y_1, \ldots, y_{t+1}|\mathcal{M}_1)}{p(y_1, \ldots, y_{t+1}|\mathcal{M}_0)},$$

where $p(y_1,\dots,y_{t+1}|\mathcal{M}_i)=\prod_{j=1}^{t+1}p(y_j|y^{j-1},\mathcal{M}_i)$, for either model.

MODEL MONITORING.

Step 1. Compute the predictive using

$$p^N(y_{t+1}|y^t)=\frac{1}{N}\sum_{i=1}^{N}p(y_{t+1}|(x_t,\theta)^{(i)}).$$

Step 2. Compute the marginal likelihood

$$p^N(y_1,\dots,y_{t+1})=\prod_{j=1}^{t+1}p^N(y_{j+1}|y^j).$$

An important advantage of PL over MCMC schemes is that it directly provides the filtered joint posteriors $p(x_t,\theta|y^t)$ and, hence, $p(y_{t+1}|y^t)$, whereas MCMC would have to be repeated T times to make that available.

4. CONDITIONAL DYNAMIC LINEAR MODELS

We now explicitly derive our PL algorithm in a class of conditional dynamic linear models which are an extension of the models considered in West and Harrison (1997). This consists of a vast class of models that embeds many of the commonly used dynamic models. MCMC via Forward-filtering Backward-sampling (Carter and Kohn, 1994; Frühwirth-Schnatter, 1994) or mixture Kalman filtering (MKF) (Chen and Liu, 2000) are the current methods of use for the estimation of these models. As an approach for filtering, PL has a number of advantages. First, our algorithm is more efficient, as it is a perfectly-adapted filter. Second, we extend MKF by including learning about fixed parameters and smoothing for states.

The conditional DLM defined by the observation and evolution equations takes the form of a linear system conditional on an auxiliary state λ_{t+1},

$$(y_{t+1}|x_{t+1},\lambda_{t+1},\theta)\sim N(F_{\lambda_{t+1}}x_{t+1},V_{\lambda_{t+1}}),$$
$$(x_{t+1}|x_t,\lambda_{t+1},\theta)\sim N(G_{\lambda_{t+1}}x_t,W_{\lambda_{t+1}}),$$

with θ containing F's, G's, V's and W's. The marginal distribution of observation error and state shock distribution are any combination of normal, scale mixture of normals or discrete mixture of normals depending on the specification of the distribution on the auxiliary state variable $p(\lambda_{t+1}|\theta)$, so that

$$p(y_{t+1}|x_{t+1},\theta)=\int f_N(y_{t+1};F_{\lambda_{t+1}}x_{t+1},V_{\lambda_{t+1}})\cdot p(\lambda_{t+1}|\theta)\,d\lambda_{t+1}.$$

Extensions to hidden Markov specifications where λ_{t+1} evolves according to $p(\lambda_{t+1}|\lambda_t,\theta)$ are straightforward and are discussed in Example 2 below.

4.1 Particle Learning in CDLM

In CDLMs the state filtering and parameter learning problem is equivalent to a filtering problem for the joint distribution of their respective sufficient statistics. This is a direct result of the factorization of the full joint

$$p(x_{t+1},\theta,\lambda_{t+1},s_{t+1},s^x_{t+1}|y^{t+1})$$

as a sequence of conditional distributions

$$p(\theta|s_{t+1})p(x_{t+1}|s^x_{t+1},\lambda_{t+1})p(\lambda_{t+1},s_{t+1},s^x_{t+1}|y^{t+1}).$$

Here the conditional sufficient statistics for states (s^x_t) and parameters (s_t) satisfy deterministic updating rules

$$s^x_{t+1}=\mathcal{K}(s^x_t,\theta,\lambda_{t+1},y_{t+1}), \tag{4.1}$$
$$s_{t+1}=\mathcal{S}(s_t,x_{t+1},\lambda_{t+1},y_{t+1}), \tag{4.2}$$

where $\mathcal{K}(\cdot)$ denotes the Kalman filter recursions and $\mathcal{S}(\cdot)$ our recursive update of the sufficient statistics. More specifically, define $s^x_t=(m_t,C_t)$ as Kalman filter first and second moments at time t. Conditional on θ, we then have

$$(x_{t+1}|s^x_{t+1},\lambda_{t+1},\theta,)\sim N(a_{t+1},R_{t+1}),$$

where $a_{t+1}=G_{\lambda_{t+1}}m_t$ and $R_{t+1}=G_{\lambda_{t+1}}C_tG'_{\lambda_{t+1}}+W_{\lambda_{t+1}}$. Updating state sufficient statistics (m_{t+1},C_{t+1}) is achieved by

$$m_{t+1}=G_{\lambda_{t+1}}m_t+A_{t+1}(y_{t+1}-e_t), \tag{4.3}$$
$$C^{-1}_{t+1}=R^{-1}_{t+1}+F'_{\lambda_{t+1}}F_{\lambda_{t+1}}V^{-1}_{\lambda_{t+1}}, \tag{4.4}$$

with Kalman gain matrix $A_{t+1}=R_{t+1}F_{\lambda_{t+1}}Q^{-1}_{t+1}$, $e_t=F_{\lambda_{t+1}}G_{\lambda_{t+1}}m_t$, and $Q_{t+1}=F_{\lambda_{t+1}}R_{t+1}F_{\lambda_{t+1}}+V_{\lambda_{t+1}}$.

We are now ready to define the PL scheme for the CDLMs. First, assume that the auxiliary state variable is discrete with $\lambda_{t+1}\sim p(\lambda_{t+1}|\lambda_t,\theta)$. We start, at time t, with a particle approximation for the joint posterior of $(x_t,\lambda_t,s_t,s^x_t,\theta|y^t)$. Then we propagate to $t+1$ by first resampling the current particles with weights proportional to the predictive

$p(y_{t+1}|(\theta, s_t^x))$. This provides a particle approximation to $p(x_t, \theta, \lambda_t, s_t, s_t^x|y^{t+1})$, the smoothing distribution. New states λ_{t+1} and x_{t+1} are then propagated through the conditional posterior distributions $p(\lambda_{t+1}|\lambda_t, \theta, y_{t+1})$ and $p(x_{t+1}|\lambda_{t+1}, x_t, \theta, y_{t+1})$. Finally, the conditional sufficient statistics are updated according to (4.1) and (4.2) and new samples for θ are obtained from $p(\theta|s_{t+1})$. Notice that in the conditional dynamic linear models all the above densities are available for evaluation and sampling. For instance, the predictive is computed via

$$p(y_{t+1}|(\lambda_t, s_t^x, \theta)^{(i)}) = \sum_{\lambda_{t+1}} p(y_{t+1}|\lambda_{t+1}, (s_t^x, \theta)^{(i)}) \cdot p(\lambda_{t+1}|\lambda_t, \theta),$$

where the inner predictive distribution is given by

$$p(y_{t+1}|\lambda_{t+1}, s_t^x, \theta) = \int p(y_{t+1}|x_{t+1}, \lambda_{t+1}, \theta) \cdot p(x_{t+1}|s_t^x, \theta)\, dx_{t+1}.$$

Starting with particle set $\{(x_0, \theta, \lambda_0, s_0, s_0^x)^{(i)}, i = 1, \ldots, N\}$ at time $t = 0$, the above discussion can be summarized in the PL Algorithm 1. In the general case where the auxiliary state variable λ_t is continuous, it might not be possible to integrate out λ_{t+1} form the predictive in step 1. We extend the above scheme by adding to the current particle set a propagated particle $\lambda_{t+1} \sim p(\lambda_{t+1}|(\lambda_t, \theta)^{(i)})$ and define the PL Algorithm 2.

Both algorithms can be combined with the backward propagation scheme of Section 3.3 to provide a full draw from the marginal posterior distribution for all the states given the data, namely, the smoothing distribution $p(x_1, \ldots, x_T|y^T)$.

ALGORITHM 1 (CDLM).

Step 1 (Resample). $\tilde{z}_t^{(i)}$ from $z_t^{(i)} = (\lambda_t, s_t^x, \theta)^{(i)}$ with weights

$$w_{t+1}^{(i)} \propto p(y_{t+1}|(\lambda_t, s_t^x, \theta)^{(i)}).$$

Step 2 (Propagate). States

$$\lambda_{t+1}^{(i)} \sim p(\lambda_{t+1}|(\tilde{\lambda}_t, \tilde{\theta})^{(i)}, y_{t+1}),$$
$$x_{t+1}^{(i)} \sim p(x_{t+1}|(\tilde{x}_t, \tilde{\theta})^{(i)}, \lambda_{t+1}^{(i)}, y_{t+1}).$$

Step 3 (Propagate). Sufficient statistics

$$s_{t+1}^{x(i)} = \mathcal{K}(\tilde{s}_t^{x(i)}, \lambda_{t+1}^{(i)}, \tilde{\theta}^{(i)}, y_{t+1}),$$
$$s_{t+1}^{(i)} = \mathcal{S}(\tilde{s}_t^{(i)}, x_{t+1}^{(i)}, \lambda_{t+1}^{(i)}, \tilde{\theta}^{(i)}, y_{t+1}).$$

Step 4 (Propagate). Parameters $\theta^{(i)} \sim p(\theta|s_{t+1}^{(i)})$.

EXAMPLE 2 (Dynamic factor model with time-varying loadings). Consider data $y_t = (y_{t1}, y_{t2})'$, $t = 1, \ldots, T$, following a dynamic factor model with time-varying loadings driven by a discrete latent state λ_t with possible values $\{1, 2\}$. Specifically, we have

$$(y_{t+1}|x_{t+1}, \lambda_{t+1}, \theta) \sim N(\beta_{t+1} x_{t+1}, \sigma^2 I_2),$$
$$(x_{t+1}|x_t, \lambda_{t+1}, \theta) \sim N(x_t, \sigma_x^2),$$

with time-varying loadings $\beta_{t+1} = (1, \beta_{\lambda_{t+1}})'$ and initial state distribution $x_0 \sim N(m_0, C_0)$. The jumps in the factor loadings are driven by a Markov switching process $(\lambda_{t+1}|\lambda_t, \theta)$, whose transition matrix Π has diagonal elements $\Pr(\lambda_{t+1} = 1|\lambda_t = 1, \theta) = p$ and $\Pr(\lambda_{t+1} = 2|\lambda_t = 2, \theta) = q$. The parameters are $\theta = (\beta_1, \beta_2, \sigma^2, \tau^2, p, q)'$. See Carvalho and Lopes (2007) for related Markov switching models.

We are able to marginalize over both (x_{t+1}, λ_{t+1}) by using state sufficient statistics $s_t^x = (m_t, C_t)$ as particles. From the Kalman filter recursions we know that $p(x_t|\lambda^t, \theta, y^t) \sim N(m_t, C_t)$. The mapping for state sufficient statistics $(m_{t+1}, C_{t+1}) = \mathcal{K}(m_t, C_t, \lambda_{t+1}, \theta, y_{t+1})$ is given by the one-step Kalman update as in (4.3) and (4.4). The prior distributions are conditionally conjugate where $(\beta_i|\sigma^2) \sim N(b_{i0}, \sigma^2 B_{i0})$ for $i = 1, 2$, $\sigma^2 \sim \text{IG}(\nu_{00}/2, d_{00}/2)$ and $\tau^2 \sim \text{IG}(\nu_{10}/2, d_{10}/2)$. For the transition probabilities, we assume that $p \sim \text{Beta}(p_1, p_2)$ and $q \sim \text{Beta}(q_1, q_2)$. Assume that, at time t, we have particles $\{(x_t, \theta, \lambda_t, s_t^x, s_t)^{(i)}\}_{i=1}^N$, for $i = 1, \ldots, N$, approximating $p(x_t, \theta, \lambda_t, s_t^x, s_t|y^t)$. The PL algorithm can be described through the following steps:

1. *Resampling*: Draw an index $k^i \sim \text{Mult}(w_t^{(1)}, \ldots, w_t^{(N)})$ with weights $w_t^{(i)} \propto p(y_{t+1}|(m_t, C_t, \lambda_t, \theta)^{(k^i)})$ where
$$p(y_{t+1}|s_t^x, \lambda_t, \theta) = \sum_{\lambda_{t+1}=1}^{2} f_N(y_{t+1}; a, b) p(\lambda_{t+1}|\lambda_t, \theta),$$
where $f_N(x; a, b)$ denotes the density of the normal distribution with mean a and variance b and evaluation at the point x. Here $a = \beta_{t+1} m_t$ and $b = (C_t + \tau^2)\beta_{t+1}\beta_{t+1}' + \sigma^2 I_2$.
2. *Propagating state* λ: Draw $\lambda_{t+1}^{(i)}$ from $p(\lambda_{t+1}|(s_t^x, \lambda_t, \theta)^{(k^i)}, y_{t+1})$:
$$p(\lambda_{t+1}|s_t^x, \lambda_t, \theta, y_{t+1}) \propto f_N(y_{t+1}; \beta_{t+1} m_t, (C_t + \tau^2)\beta_{t+1}\beta_{t+1}' + \sigma^2 I_2) \cdot p(\lambda_{t+1}|\lambda_t, \theta).$$

3. *Propagating state x*: Draw $x_{t+1}^{(i)}$ from $p(x_{t+1}|\lambda_{t+1}^{(i)}, (s_t^x,\theta)^{(k^i)}, y_{t+1})$.
4. *Propagating sufficient statistics for states*: The Kalman filter recursions yield

$$m_{t+1} = m_t + A_{t+1}(y_{t+1} - \beta_{t+1} m_t),$$
$$C_{t+1} = C_t + \tau^2 - A_{t+1} Q_{t+1}^{-1} A_{t+1}',$$

where $Q_{t+1} = (C_t + \tau^2)\beta_{t+1}\beta_{t+1} + \sigma^2 I_2$ and $A_{t+1} = (C_t + \tau^2) Q_{t+1}^{-1} \beta_{t+1}$.
5. *Propagating sufficient statistics for parameters*: The conditional posterior $p(\theta|s_t)$, for $i = 1, 2$, is decomposed into

$$p(\beta_i|\sigma^2, s_{t+1}) \sim N(b_{i,t+1}, \sigma^2 B_{i,t+1}),$$
$$p(\sigma^2|s_{t+1}) \sim \mathrm{IG}(\nu_{0,t+1}/2, d_{0,t+1}/2t),$$
$$p(\tau^2|s_{t+1}) \sim \mathrm{IG}(\nu_{1,t+1}/2, d_{1,t+1}/2),$$
$$p(p|s_{t+1}) \sim \mathrm{Beta}(p_{1,t+1}, p_{2,t+1}),$$
$$p(q|s_{t+1}) \sim \mathrm{Beta}(q_{1,t+1}, q_{2,t+1}),$$

with $B_{i,t+1}^{-1} = B_{it}^{-1} + x_{t+1}^2 \mathbb{I}_{\lambda_{t+1}=i}$, $b_{i,t+1} = B_{i,t+1} \cdot (B_{it}^{-1} b_{it} + x_t y_{t2} \mathbb{I}_{\lambda_{t+1}=i})$ and $\nu_{i,t+1} = \nu_{i,t} + 1$, for $i = 1, 2$, $d_{1,t+1} = d_{1t} + (x_{t+1} - x_t)^2$, $p_{1,t+1} = p_{1t} + \mathbb{I}_{\lambda_t=1,\lambda_{t+1}=1}$, $p_{2,t+1} = p_{2t} + \mathbb{I}_{\lambda_t=1,\lambda_{t+1}=2}$, $q_{1,t+1} = q_{1t} + \mathbb{I}_{\lambda_t=2,\lambda_{t+1}=2}$ $q_{2,t+1} = q_{2t} + \mathbb{I}_{\lambda_t=2,\lambda_{t+1}=1}$ and $d_{0,t+1} = d_{0t} + \sum_{j=1}^{2}[(y_{t+1,2} - b_{j,t+1}x_{t+1})y_{t+1,2} + b_{j,t+1}B_{j0}^{-1} + (y_{t+1,1} - x_{t+1})^2]\mathbb{I}_{\lambda_{t+1}=j}$.

Figures 1 and 2 illustrate the performance of the PL algorithm. The first panel of Figure 1 displays the true underlying λ process along with filtered and smoothed estimates, whereas the second panel presents the same information for the common factor. Figure 2 provides the sequential parameter learning plots.

ALGORITHM 2 (Auxiliary state CDLM). Let $z_t = (\lambda_{t+1}, x_t, s_t^x, \theta)$.

Step 0 (Propagate). $\lambda_t^{(i)}$ to $\lambda_{t+1}^{(i)}$ via $\lambda_{t+1}^{(i)} \sim p(\lambda_{t+1}|(\lambda_t,\theta)^{(i)})$.

Step 1 (Resample). $\tilde{z}_t^{(i)}$ from $z_t^{(i)}$ with weights $w_{t+1}^{(i)} \propto p(y_{t+1}|\tilde{z}_t^{(i)})$.

Step 2 (Propagate). $\tilde{x}_t^{(i)}$ to $x_{t+1}^{(i)}$ via $p(x_{t+1}|\tilde{z}_t^{(i)}, y_{t+1})$.

Step 3 (Propagate). Sufficient statistics as in PL.

Step 4 (Propagate). Parameters as in PL.

5. NONLINEAR FILTERING AND LEARNING

We now extend our PL filter to a general class of nonlinear state space models, namely, the conditional Gaussian dynamic model (CGDM). This class generalizes conditional dynamic linear models by allowing nonlinear evolution equations. In this context we take advantage of most efficiency gains of PL, as we are still able to follow the resample/propagate logic and filter sufficient statistics for θ. Consider a conditional Gaussian state space model with nonlinear evolution equation,

$$(y_{t+1}|x_{t+1}, \lambda_{t+1}, \theta) \sim N(F_{\lambda_{t+1}} x_{t+1}, V_{\lambda_{t+1}}), \tag{5.1}$$
$$(x_{t+1}|x_t, \lambda_{t+1}, \theta) \sim N(G_{\lambda_{t+1}} h(x_t), W_{\lambda_{t+1}}), \tag{5.2}$$

where $h(\cdot)$ is a given nonlinear function and, again, θ contains F's, G's, V's and W's. Due to the nonlinearity in the evolution, we are no longer able to work with state sufficient statistics s_t^x, but we are still able to evaluate the predictive $p(y_{t+1}|x_t, \lambda_t, \theta)$. In general, take as the particle set the following: $\{(x_t, \theta, \lambda_t, s_t)^{(i)}, i = 1, \ldots, N\}$. For discrete λ we can define the following algorithm:

ALGORITHM 3 (CGDM).

Step 1 (Resample). $\tilde{z}_t^{(i)}$ from $z_t^{(i)} = (x_t, \lambda_t, \theta)^{(i)}$ with weights

$$w_t^{(i)} \propto p(y_{t+1}|(x_t, \lambda_t, \theta)^{(i)}).$$

Step 2 (Propagate). States

$$\lambda_{t+1}^{(i)} \sim p(\lambda_{t+1}|(\tilde{\lambda}_t, \tilde{\theta})^{(i)}, y_{t+1}),$$
$$x_{t+1}^{(i)} \sim p(x_{t+1}|(\tilde{x}_t, \tilde{\theta})^{(i)}, \lambda_{t+1}^{(i)}, y_{t+1}).$$

Step 3 (Propagate). Parameter sufficient statistics as in Algorithm 1.

Step 4 (Propagate). Parameters as in PL.

When λ is continuous, propagate $\lambda_{t+1}^{(i)}$ from $p(\lambda_{t+1}|(\lambda_t,\theta)^{(i)})$, for $i = 1, \ldots, N$, then we resample the particle $(x_t, \lambda_{t+1}, \theta, s_t)^{(i)}$ with the appropriate predictive distribution $p(y_{t+1}|(x_t, \lambda_{t+1}, \theta)^{(i)})$ as in Algorithm 2. Finally, it is straightforward to extend the backward smoothing strategy of Section 3.3 to obtain samples from $p(x^T|y^T)$.

EXAMPLE 3 (Heavy-tailed nonlinear state space model). Consider the following non-Gaussian and nonlinear state space model

$$(y_{t+1}|x_{t+1}, \lambda_{t+1}, \theta) \sim N(x_{t+1}, \lambda_{t+1}\sigma^2),$$
$$(x_{t+1}|x_t, \lambda_{t+1}, \theta) \sim N(\beta h(x_t), \sigma_x^2),$$

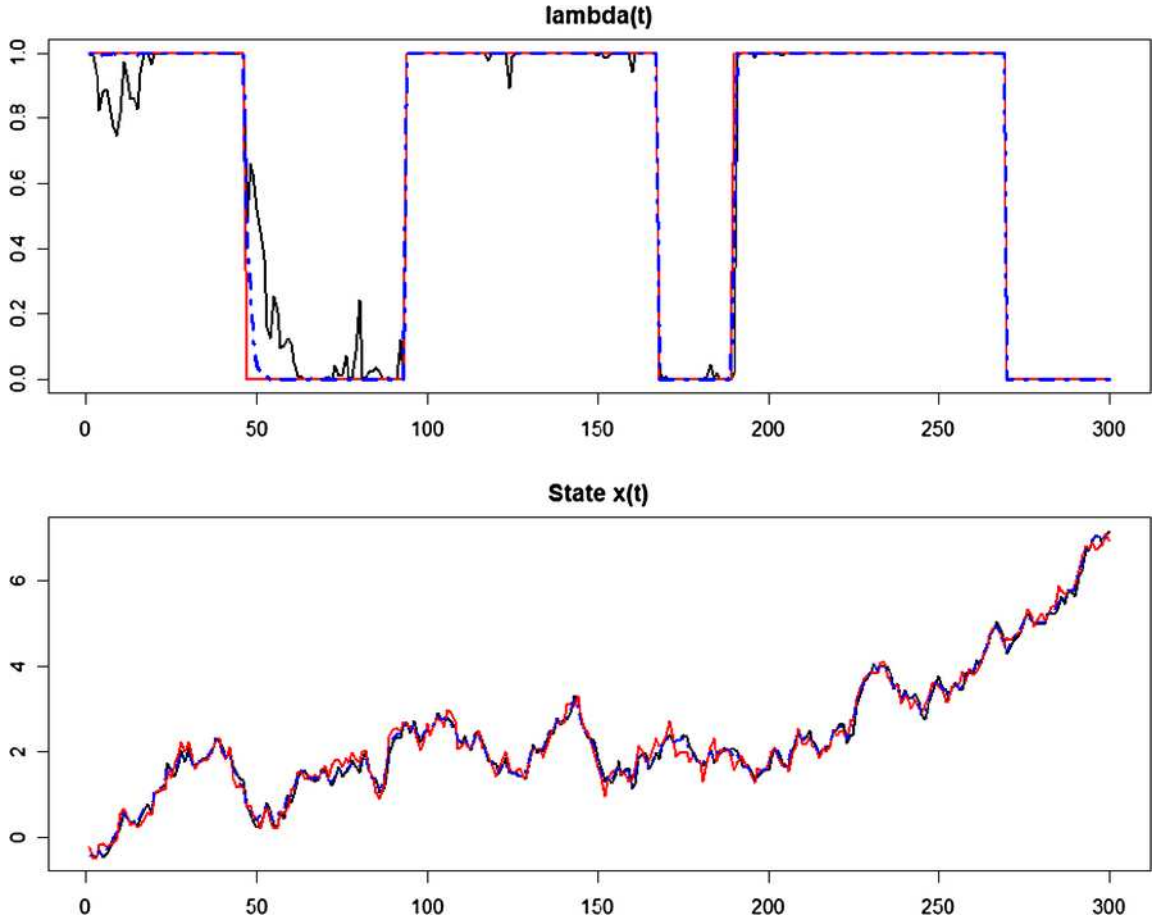

FIG. 1. *Dynamic factor model (state learning). Top panel: True value of* λ_t *(red line),* $\Pr(\lambda_t = 1|y^t)$ *(black line) and* $\Pr(\lambda_t = 1|y^T)$ *(blue line). Bottom panel: True value of* x_t *(red line),* $E(x_t|y^t)$ *(black line) and* $E(x_t|y^T)$ *(blue line).*

where $\theta = (\beta, \sigma^2, \tau^2)$, $h(x_t) = x_t/(1+x_t^2)$ and $\lambda_{t+1} \sim \text{IG}(\nu/2, \nu/2)$, for known ν. Therefore, the distribution of $(y_{t+1}|x_{t+1}, \theta) \sim t_\nu(x_{t+1}, \sigma^2)$, that is, a t-Student with ν degrees of freedom.

The particle learning algorithm works as follows. Let the particle set $\{(x_t, \theta, \lambda_{t+1}, s_t)^{(i)}\}_{i=1}^N$ approximate $p(x_t, \theta, \lambda_{t+1}, s_t|y^t)$. For anygiven time $t = 0, \ldots, T-1$ and $i = 1, \ldots, N$, we first draw an index $k^i \sim \text{Mult}(w_t^{(1)}, \ldots, w_t^{(N)})$, with $w_t^{(j)} \propto p(y_{t+1}|(x_t, \lambda_{t+1}, \theta)^{(j)})$, $j = 1, \ldots, N$, and $p(y_{t+1}|x_t, \lambda_{t+1}, \theta) = f_N(y_{t+1}; \beta h(x_t), \lambda_{t+1}\sigma^2 + \tau^2)$. Then, we draw a new state $x_{t+1}^{(i)} \sim p(x_{t+1}|(\lambda_{t+1}, x_t, \theta)^{(k^i)}, y_{t+1}) \equiv f_N(x_{t+1}; \mu_{t+1}^{(i)}, V_{t+1}^{(i)})$, where $\mu_{t+1} = V_{t+1}(\lambda_{t+1}^{-1}\sigma^{-2}y_{t+1} + \tau^{-2} \cdot \beta h(x_t))$ and $V_{t+1}^{-1} = \lambda_{t+1}^{-1}\sigma^{-2} + \tau^{-2}$. Finally, similar to Example 1, posterior parameter learning for $\theta = (\beta, \sigma^2, \tau^2)$ follows directly from a conditionally normal-inverse gamma update. Figure 3 illustrates the above PL algorithm in a simulated example where $\beta = 0.9$, $\sigma^2 = 0.04$ and $\sigma_x^2 = 0.01$. The algorithm uncovers the true parameters very efficiently in a sequential fashion. In Section 6.1 we revisit this example to compare the performances of PL, MCMC (Carlin, Polson and Stoffer, 1992) and the benchmark particle filter with parameter learning (Liu and West, 2001).

6. COMPARING PARTICLE LEARNING TO EXISTING METHODS

We now present a series of examples that illustrate the performance of PL benchmarked by commonly used alternatives.

EXAMPLE 4 (State sufficient statistics). In this first simulation exercise we revisit the local level model of Example 1 in order to compare PL to its version that takes advantage of state sufficient statistics, that is, by marginalizing the latent states. The main goal is to study the Monte Carlo error of the two filters. We simulated a time series of length

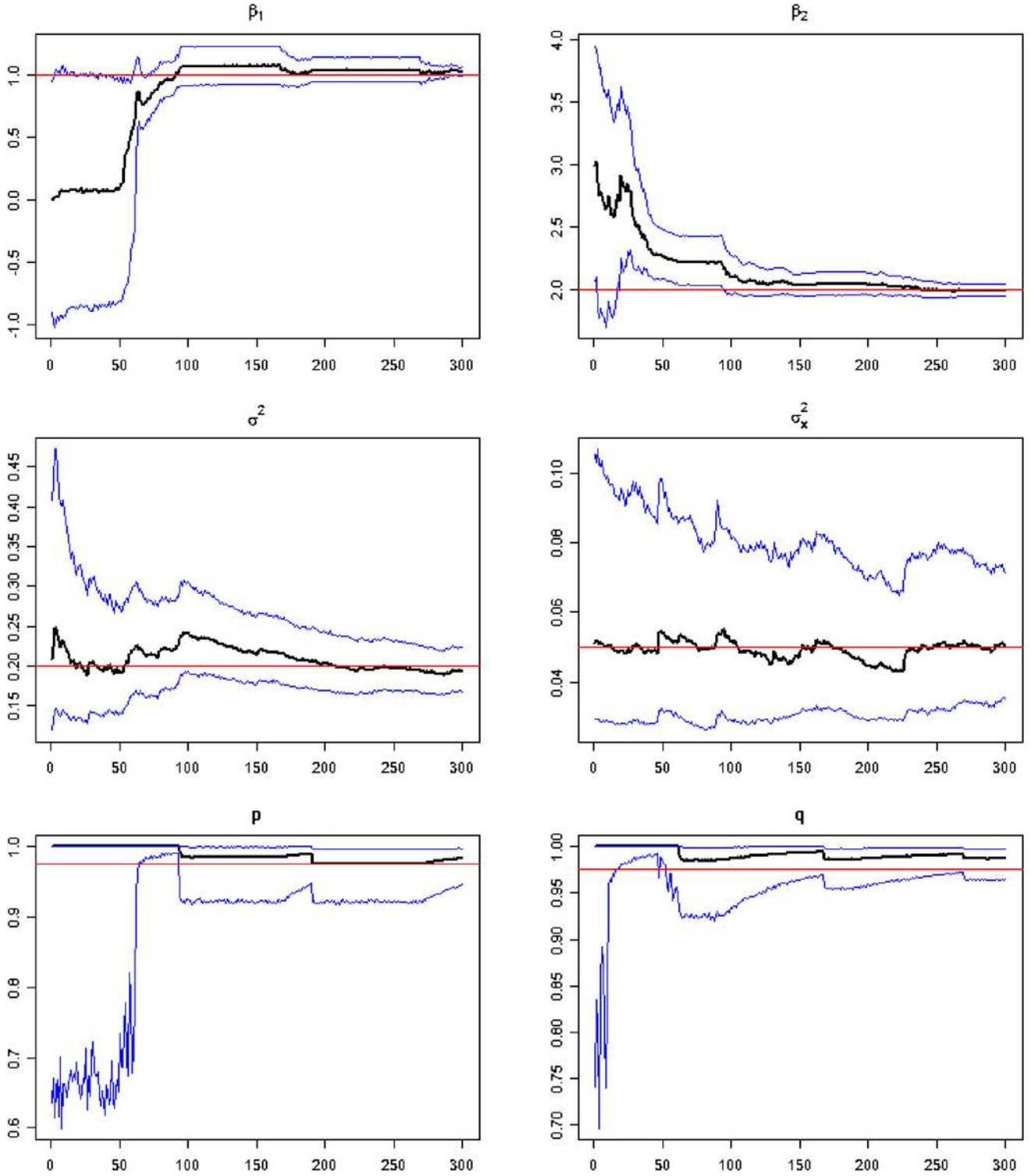

FIG. 2. *Dynamic factor model (parameter learning). Sequential posterior median (black line) and posterior 95% credibility intervals (blue lines) for model parameters β_1, β_2, σ^2, τ^2, p and q. True values are the red lines.*

$T = 100$ with $\sigma^2 = 1$, $\tau^2 = 0.1$ and $x_0 = 0_p$. The prior distributions are $\sigma^2 \sim \text{IG}(5, 4)$, $\tau^2 \sim \text{IG}(5, 0.4)$ and $x_0 \sim N(0, 10)$. We run two filters: one with sequential learning for x_t, σ^2 and τ^2 (we call it simply *PL*), and the other with sequential learning for state sufficient statistics, σ^2 and τ^2 (we call it *PLsuff*). In

both cases, the particle filters are based on either one long particle set of size $N = 100{,}000$ (we call it *Long*) or 20 short particle sets of size $N = 5000$ (we call it *Short*). The results are in Figures 4 to 6. Figure 4 shows that the differences between *PL* and *PLsuff* dissipate for fairly large N. However, when N is small *PLsuff* has smaller Monte Carlo error and is less biased than *PL*, particularly when estimating σ^2 and τ^2 (see Figure 5). Similar findings appear in Figure 6 where the mean square errors of the quantiles from the 20 *Short* runs are compared to those from the *Long PLsuff* run.

EXAMPLE 5 (Resample–propagate or propagate–resample?). In this second simulation exercise we continue focusing in the local level model of Example 1 to compare PL to three other particle filters: the bootstrap filter (BF), its fully adapted version (FABF), and the auxiliary particle filter (APF) (no fully adapted). BF and FABF are propagate–resample filters, while PL and APF are resample–propagate filters. The main goal is to study the Monte Carlo error of the four filters. We start with the pure case scenario, that is, with fixed parameters. We simulated 20 time series of length $T = 100$ from the local level model with parameters $\tau^2 = 0.013$, $\sigma^2 = 0.13$ and $x_0 = 0$. Therefore, the signal to noise ratio σ_x/σ equals 0.32. Other combinations were also tried and similar results were found. The prior distribution of the initial state x_0 was set at $N(0, 10)$. For each time series, we run 20 times on each of the four filters, all based on $N = 1000$ particles. We use five quantiles to compare the various filters. Let q_α^t be such that $\Pr(x_t < q_t^\alpha | y^t) = \alpha$, for $\alpha = (0.05, 0.25, 0.5, 0.75, 0.95)$. Then, the mean square er-

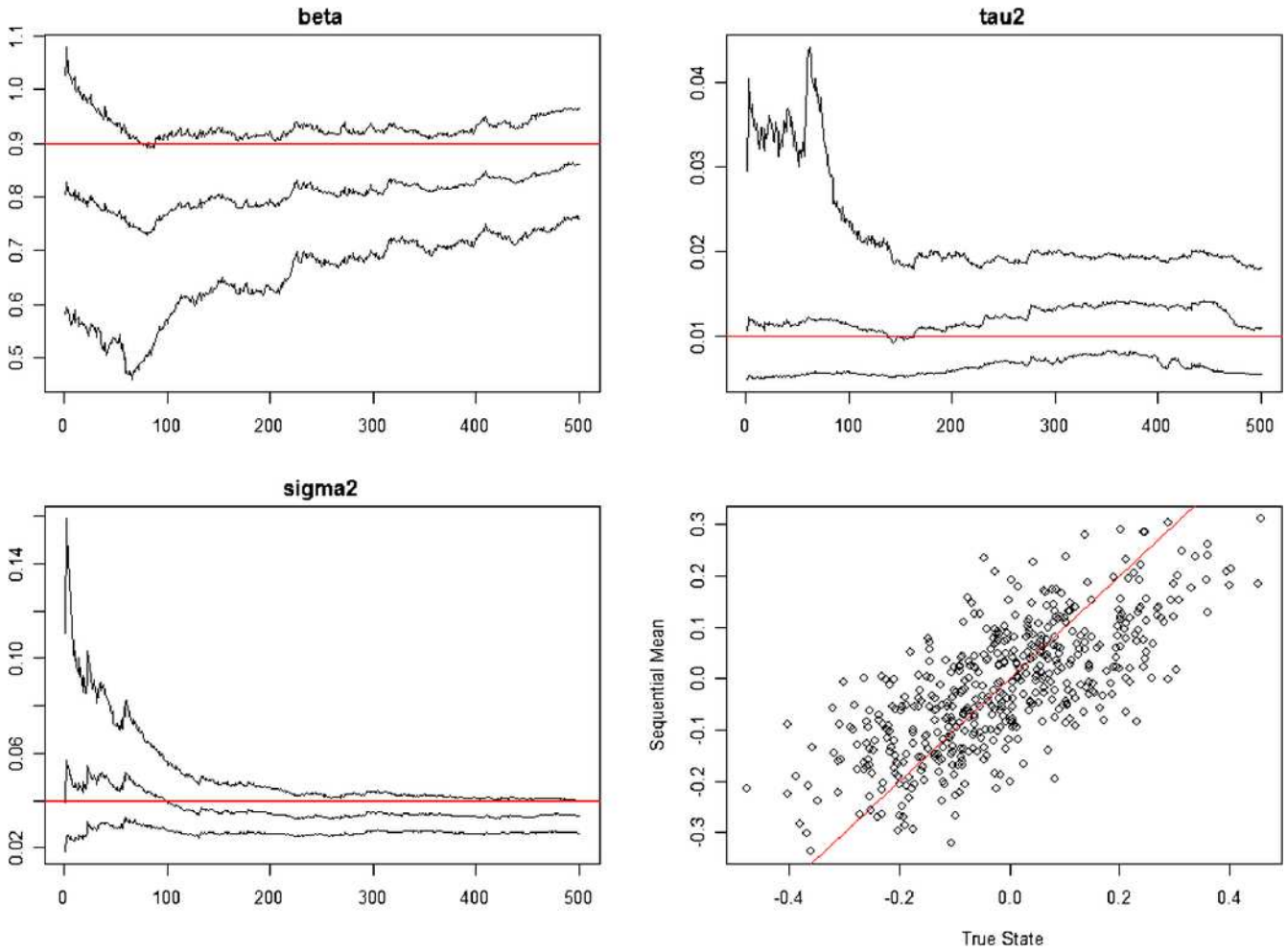

FIG. 3. *Heavy-tailed non-Gaussian, nonlinear model. Sequential posterior median and posterior 95% credibility intervals (black lines) for model parameters β, σ^2 and τ^2. True values are the red lines. The bottom right panel is the true value of x_t against $E(x_t|y^t)$.*

ror (MSE) for filter f, at time t and quantile α is

$$MSE_{t,f}^{\alpha} = \frac{1}{400}\sum_{d=1}^{20}\sum_{r=1}^{20}(q_{t,d}^{\alpha} - \hat{q}_{t,d,f,r}^{\alpha})^2,$$

where d and r index the data set and the particle filter run, respectively. We compare PL, APF and FABF via logarithm relative MSE (LRMSE), relative to the benchmark BF. Results are summarized in Figure 7. PL is uniformly better than all three alternatives. Notice that the only algorithmic difference between PL and FABF is that PL reverses the propagate–resample steps.

We now move to the parameter learning scenario, where σ^2 is still kept fixed but learning of τ^2 is performed. Three time series of length $T = 1000$ were simulated from the local level model with $x_0 = 0$ and (σ^2, τ^2) in $\{(0.1, 0.01), (0.01, 0.01), (0.01, 0.1)\}$. The independent prior distributions for x_0 and τ^2 are $x_0 \sim N(0, 1)$ and $\tau^2 \sim \mathrm{IG}(10, 9\tau_0^2)$, where τ_0^2 is the true value of τ^2 for a given time series. In all filters τ^2 is sampled offline from $p(\tau^2|s_t)$ where s_t is the vector of conditional sufficient statistics. We run the filters 100 times, all with the same seed within run, for each one of the three simulated data sets. Finally, the number of particles was set at $N = 5000$, with similar results found for smaller N, ranging from 250 to 2000 particles. Mean absolute errors (MAE) over the 100 replications are constructed by comparing quantiles of the true sequential distributions $p(x_t|y^t)$ and $p(\tau^2|y^t)$ to quantiles of the estimated sequential distributions $p^N(x_t|y^t)$ and $p^N(\tau^2|y^t)$. More specifically, for time t, a in $\{x, \tau^2\}$, α in $\{0.01, 0.50, 0.99\}$, true quantiles $q_{t,a}^{\alpha}$ and PL quantiles $\hat{q}_{t,a,r}^{\alpha}$,

$$MAE_{t,a}^{\alpha} = \frac{1}{100}\sum_{r=1}^{100}|q_{t,a}^{\alpha} - \hat{q}_{t,a,r}^{\alpha}|.$$

Across different quantiles and combinations of error variances, PL is at least as good as FABF and in many cases significantly better than BF. Results appear in Figure 8.

EXAMPLE 6 (PL versus LW). Consider once again a variation of the dynamic linear model introduced in Example 1, but now we assume complete knowledge about (σ^2, τ^2) in

$$(y_{t+1}|x_{t+1}, \beta) \sim N(x_t, \sigma^2),$$
$$(x_{t+1}|x_t, \beta) \sim N(\beta x_t, \tau^2)$$

for $t = 1, \ldots, T = 100$, $\sigma^2 = 1$, $x_1 = 0.0$ and three possible values for $\tau^2 = (0.01, 0.25, 1.00)$. So, the signal to noise ratio $\tau/\sigma = 0.1, 0.5, 1.0$. Only β and x_t are sequentially estimated and their independent prior distributions are $N(1.0, 1.0)$ and $N(0.0, 1.0)$, respectively. The particle set has length $N = 2000$ and both filters were run 50 times to study the size of the Monte Carlo error. The smoothing parameter δ of Liu and West's filter was set at $\delta = 0.95$, but fairly similar results were found for δ ranging from 0.8 to 0.99. Our findings, summarized in Figure 9, favor PL over LW uniformly across all scenarios. The discrepancy is higher when τ/σ is small, which is usually the case in state space applications.

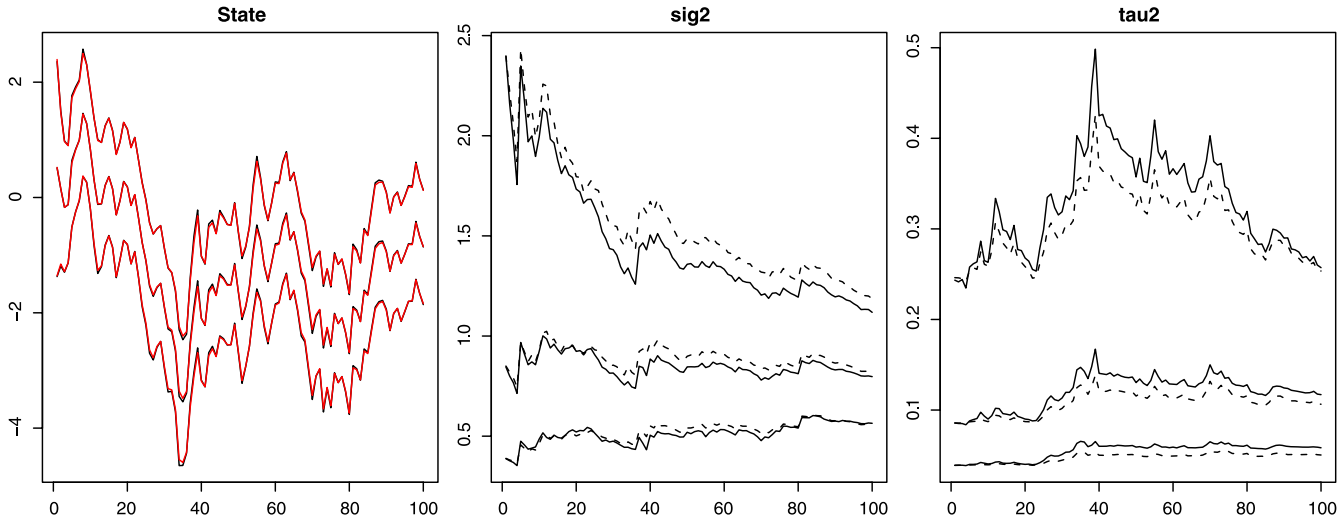

FIG. 4. *PL and PL with state sufficient statistics (long runs). Left panel—$p(x_t|y^t)$—PL (black), PLsuff (red); Middle panel—$p(\sigma^2|y^t)$—PL (solid line), PLsuff (dotted line); Right panel—$p(\tau^2|y^t)$—PL (solid line), PLsuff (dotted line).*

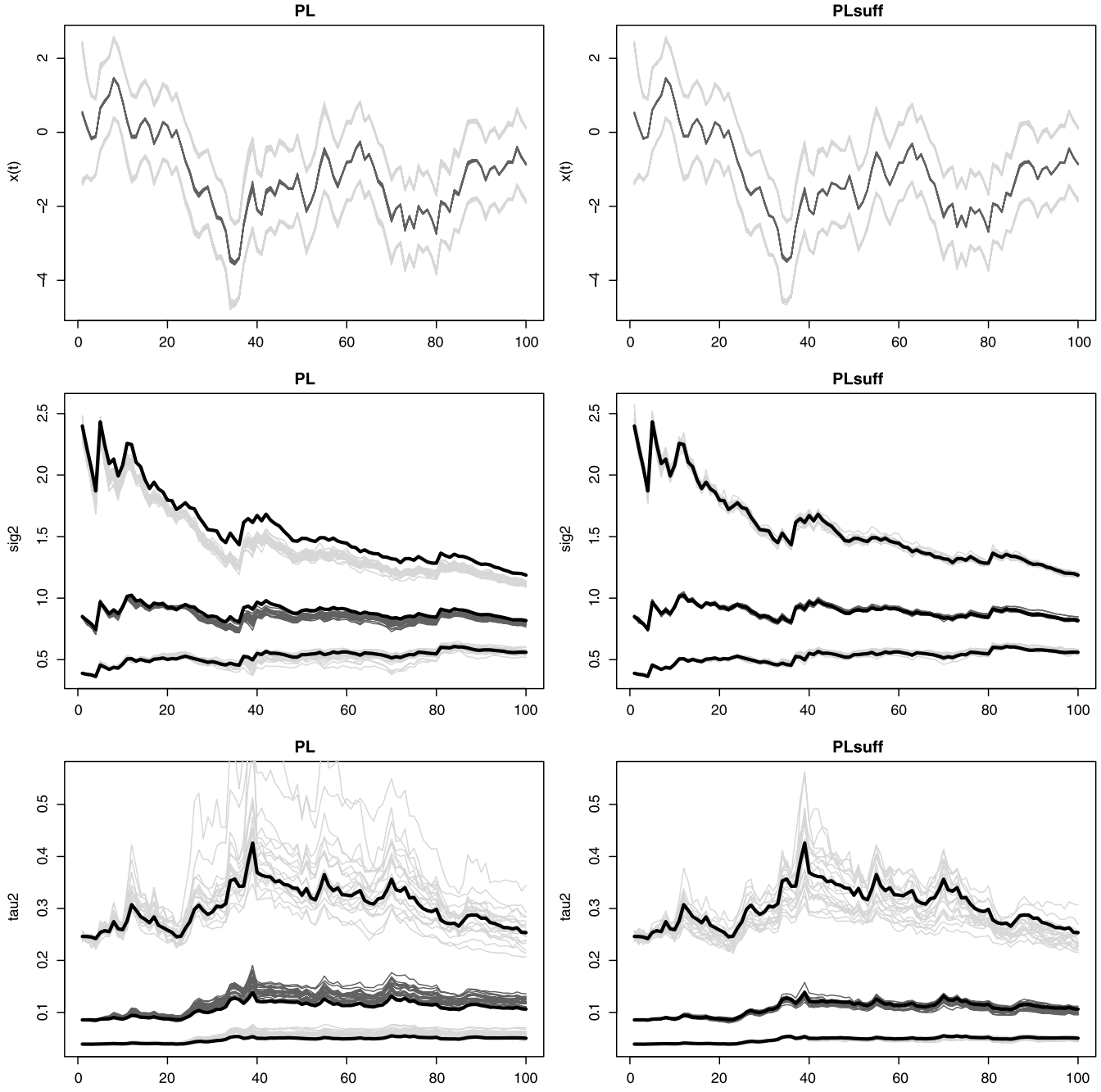

FIG. 5. *PL and PL with state sufficient statistics (20 short runs). PL runs (left columns) and PLsuff runs (right columns). One long run (black) and 20 short runs (gray); $p(x_t|y^t)$ (top row), $p(\sigma^2|y^t)$ (middle row) and $p(\tau^2|y^t)$ (bottom row).*

6.1 PL vs MCMC

PL combined with the backward smoothing algorithm (as in Section 3.3) is an alternative to MCMC methods for state space models. In general, MCMC methods (see Gamerman and Lopes, 2006) use Markov chains designed to explore the posterior distribution $p(x^T, \theta|y^T)$ of states and parameters conditional on all the information available, $y^T = (y_1, \ldots, y_T)$. For example, an MCMC strategy would have to iterate through

$$p(\theta|x^T, y^T) \quad \text{and} \quad p(x^T|\theta, y^T).$$

However, MCMC relies on the convergence of very high-dimensional Markov chains. In the purely conditional Gaussian linear models or when states are dicrete, $p(x^T|\theta, y^T)$ can be sampled in block using

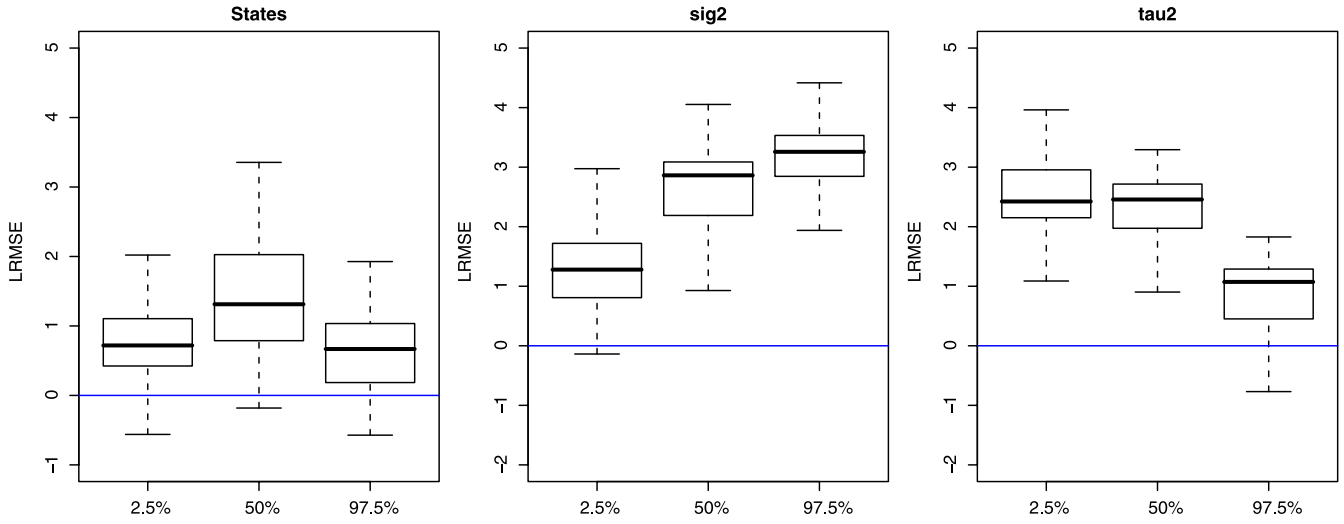

FIG. 6. *PL and PL with state sufficient statistics (mean square errors). Logarithm of the relative mean square error for three quantiles of* $p^N(x_t|y^t)$, $p^N(\sigma^2|y^t)$ *and* $p^N(\tau^2|y^t)$, *averaged across the 20* $N = 5000$ *runs. PL relative to PLsuff.*

FFBS. Even in these ideal cases, achieving convergency is far from an easy task and the computational complexity is enormous, as at each iteration one would have to filter forward and backward sample for the full state vector x^T. The particle learning algorithm presented here has two advantages: (i) it requires only one forward/backward pass through the data for all N particles and (ii) the approximation accuracy does not rely on convergence results that are virtually impossible to assess in practice (see Papaspiliopoulos and Roberts, 2008).

In the presence of nonlinearities, MCMC methods will suffer even further, as no FFBS scheme is available for the full state vector x^T. One would have to resort to univariate updates of $p(x_t|x_{(-t)}, \theta, y^T)$ as in Carlin, Polson and Stoffer (1992), where $x_{(-t)}$ is x^T without x_t. It is well known that these methods generate very "sticky" Markov chains, increasing computational complexity and slowing down convergence. PL is also attractive given the simple nature of its implementation (especially if compared to more novel hybrid methods).

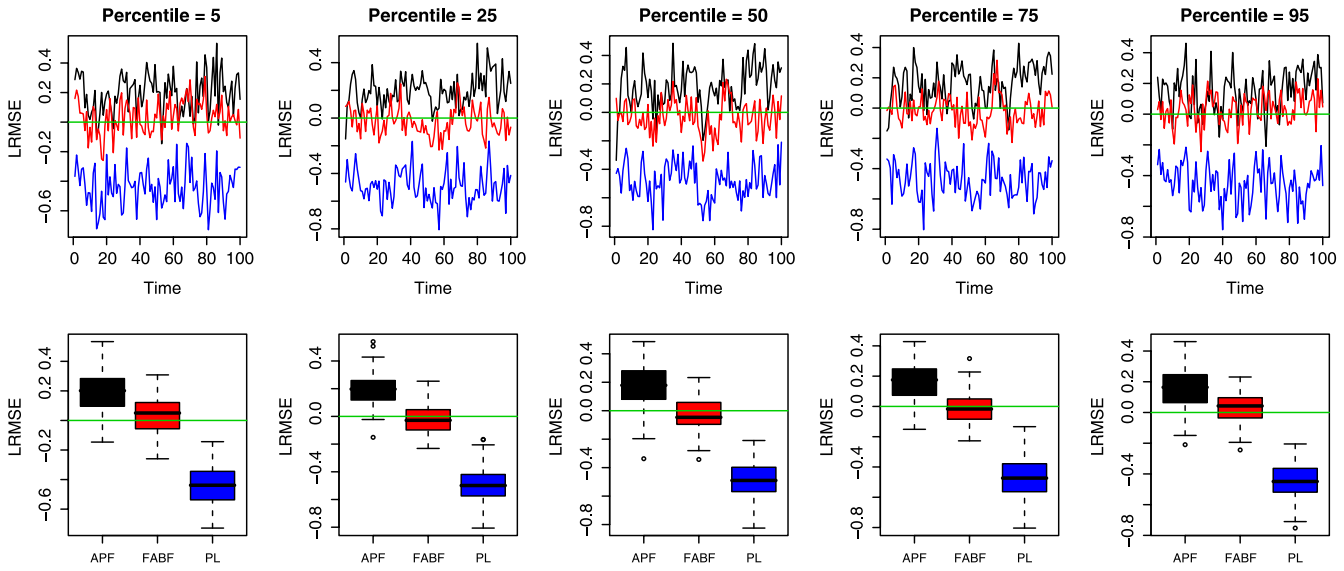

FIG. 7. *APF, FABF and PL pure filter. Logarithm of the relative mean square error for five quantiles of* $p^N(x_t|y^t)$. *MSE relative to BF. Boxplots on the second row are based on the time series plots on the first row.*

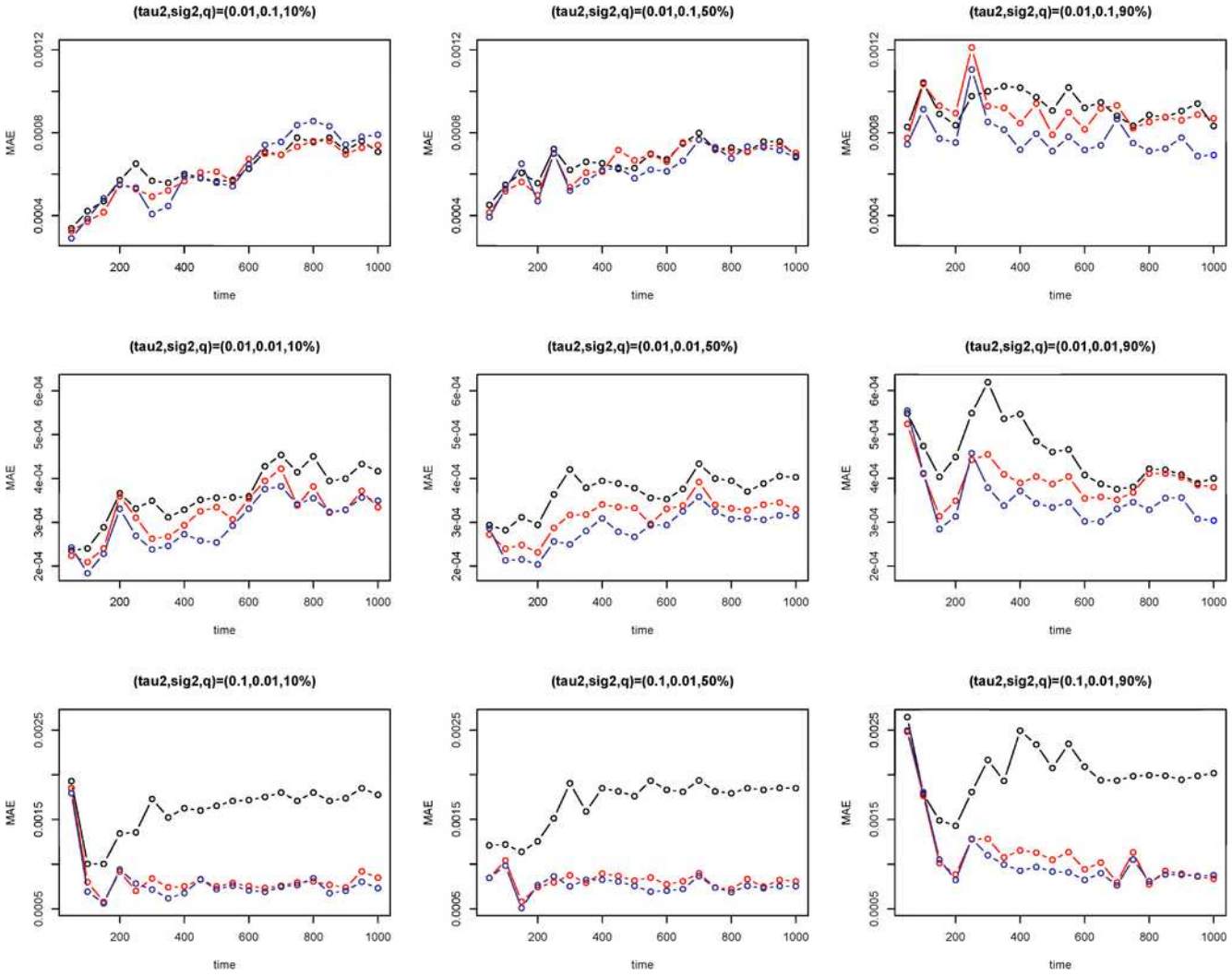

FIG. 8. *BF, FABF and PL with learning of τ^2. Mean absolute errors. BF (black), FABF (red) and PL (blue).*

EXAMPLE 7 (PL versus FFBS). We revisit the first order dynamic linear model introduced in Example 1 to compare our PL smoother and the forward-filtering, backward-sampling (FFBS) smoother. Assuming knowledge about θ, Figure 10 compares the true smoothed distributions $p(x_t|y^T)$ to approximations based on PL and on FFBS. Now, when parameter learning is introduced, PL performance is comparable to that of the FFBS when approximating $p(\sigma^2, \tau^2|y^T)$, as shown in Figure 11. We argue that, based on these empirical findings, PL and FFBS are equivalent alternatives for posterior computation. We now turn to the issue of computational cost, measured here by the running time in seconds of both schemes. Data was simulated based on $(\sigma^2, \tau^2, x_0) = (1.0, 0.5, 0.0)$. The prior distribution of x_0 is $N(0, 100)$, while σ^2 and τ^2 are kept fixed throughout this exercise. PL was based on N particles and FFBS based on $2N$ iterations, with the first M discarded. Table 1 summarizes the results. For fixed N, the (computational) costs of both PL and FFBS increase linearly with T, with FFBS twice as fast as PL. For fixed T, the cost of FFBS increases linearly with N, while the cost of PL increases exponentially with N. These findings were anticipated in Section 3.3. As expected, PL outperforms FFBS when comparing filtering times.

EXAMPLE 8 (PL versus single-move MCMC). Our final example compares PL to a single-move MCMC as in Carlin, Polson and Stoffer (1992). We consider the first order conditional Gaussian dynamic model with nonlinear state equation as defined in Example 3. The example focuses on the estimation

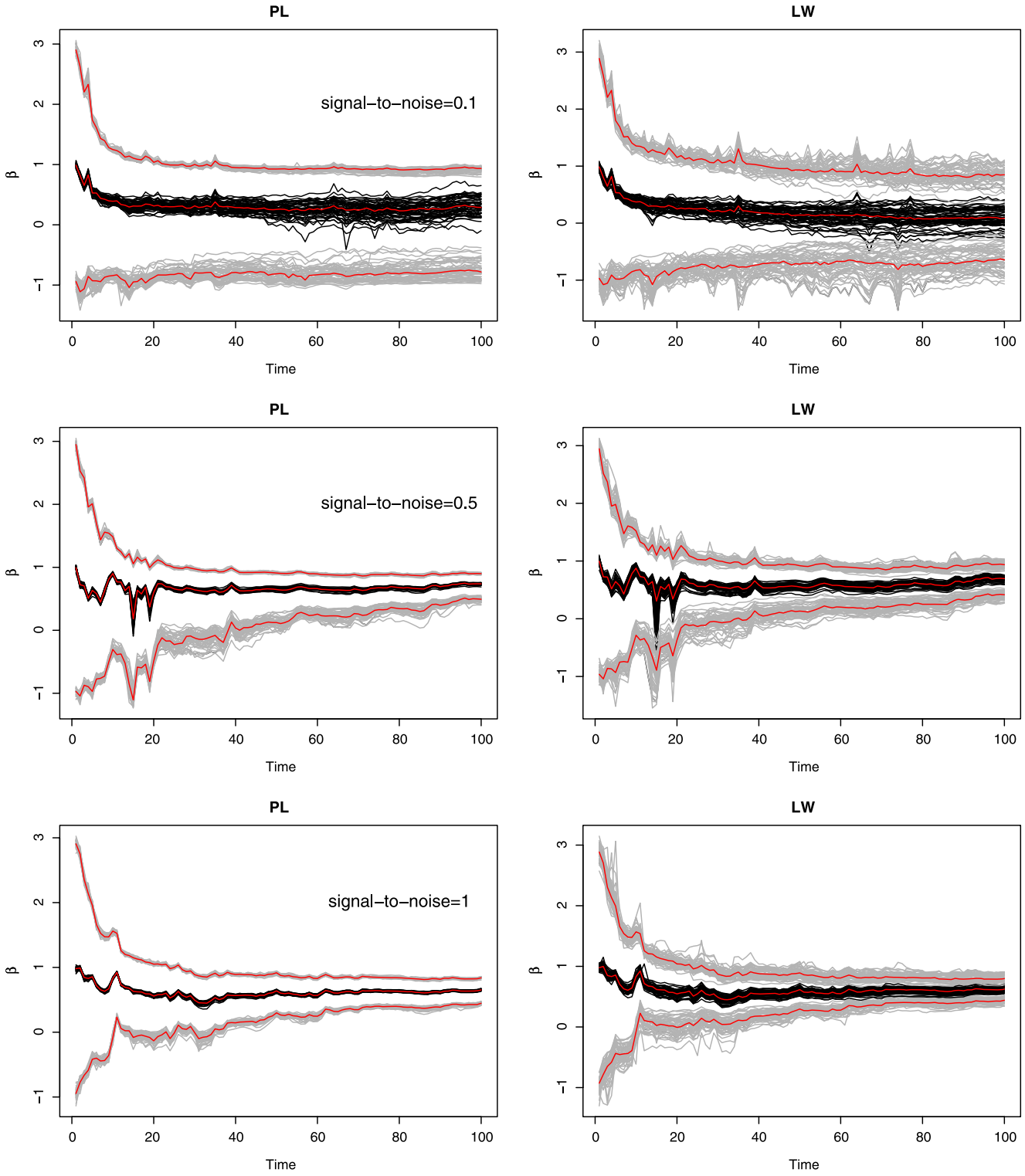

FIG. 9. *PL and LW (parameter learning). Posterior mean and 95% credibility interval from* $p(\beta|y^t)$. *Medians across the 50 runs appear in red.* $N = 2000$ *particles.* `signal-to-noise` *stands for* σ_x/σ. *In all cases,* $\sigma = 1$.

of σ^2. We generate data with different levels of signal to noise ratio and compare the performance of PL versus MCMC. Table 2 presents the results for the comparisons. Once again, PL provides significant improvements in computational time and MC variability for parameter estimation over MCMC.

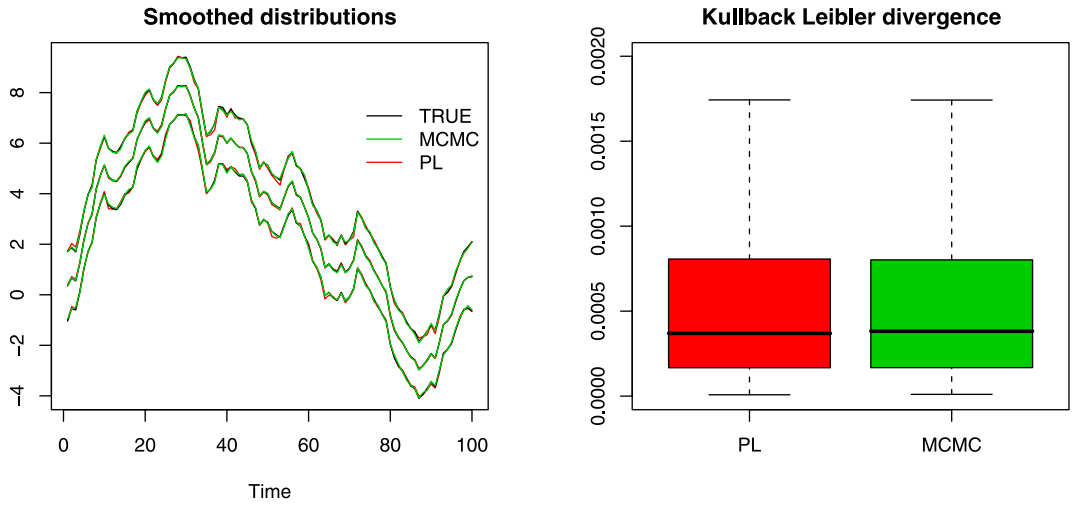

FIG. 10. *PL and FFBS (smoothed distributions). $T = 100$ simulated from a local level model with $\sigma^2 = 1$, $\tau^2 = 0.5$, $x_0 = 0$ and $x_0 \sim N(0, 100)$. PL is based on $N = 1000$ particles, while FFBS is based on $2N$ draws with the first N discarded.*

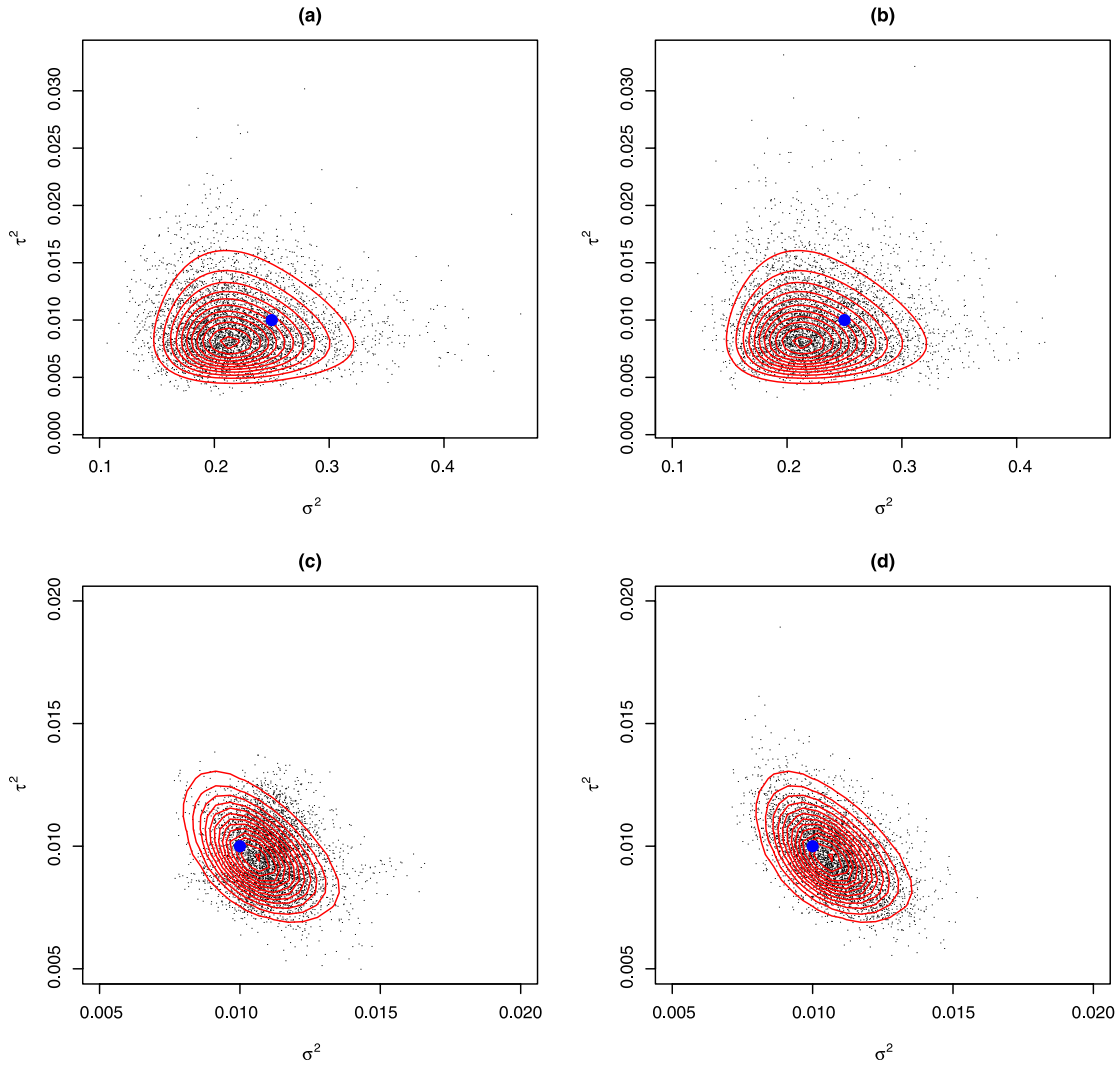

FIG. 11. *PL and FFBS (parameter learning). Contour plots for the true posterior $p(\sigma^2, \tau^2 | y^T)$ (red contours) and posterior draws from PL, panels* (a) *and* (c)*, and FFBS, panels* (b) *and* (d)*. The blue dots represent the true value of the pair (σ^2, τ^2). The sample size is $T = 50$ (top row) and $T = 500$ (bottom row).*

TABLE 1
Computing time (in seconds) of PL and FFBS for smoothing. In parenthesis are PL times for filtering

$N = 500$			$T = 100$		
T	PL	FFBS	N	PL	FFBS
200	18.8 (0.25)	9.1	500	9.3 (0.09)	4.7
500	47.7 (1.81)	23.4	1000	32.8 (0.15)	9.6
1000	93.9 (8.29)	46.1	2000	127.7 (0.34)	21.7

TABLE 2
Single-move MCMC based on 2000 draws, after 2000 burn-in. PL based on 2000 particles. Expectations are with respect to the whole data set at time T, while the true value of τ^2 is 0.01. *Numbers in parenthesis are 1000 times the standard deviation based on 20 replications of the algorithms. Time is in seconds when running our code in R version 2.8.1 on a MacBook with a 2.4 GHz processor and 4 GB MHz of memory*

T	σ^2	Time	$E(\sigma^2)$	$E(\tau^2)$
			Single-move MCMC	
50	0.2500	19.7	0.209934 (3.901)	0.011 (1.532)
	0.0100	19.3	0.009151 (0.253)	0.008 (0.545)
	0.0001	19.3	0.000097 (0.003)	0.010 (0.049)
200	0.2500	79.3	0.249059 (6.981)	0.027 (12.76)
	0.0100	79.1	0.009740 (0.305)	0.013 (1.375)
	0.0001	79.8	0.000099 (0.004)	0.011 (0.032)
			PL	
50	0.2500	0.8	0.170576 (1.633)	0.010 (0.419)
	0.0100	0.7	0.007204 (0.151)	0.008 (0.165)
	0.0001	0.6	0.000092 (0.004)	0.010 (0.058)
200	0.2500	6.5	0.262396 (6.392)	0.009 (1.332)
	0.0100	6.4	0.010615 (0.570)	0.011 (0.935)
	0.0001	6.4	0.000098 (0.010)	0.011 (0.057)

7. FINAL REMARKS

In this paper we provide particle learning tools (PL) for a large class of state space models. Our methodology incorporates sequential parameter learning, state filtering and smoothing. This provides an alternative to the popular FFBS/MCMC (Carter and Kohn, 1994) approach for conditional dynamic linear models (DLMs) and also to MCMC approaches to nonlinear non-Gaussian models. It is also a generalization of the mixture Kalman filter (MKF) approach of Chen and Liu (2000) that includes parameter learning and smoothing. The key assumption is the existence of a conditional sufficient statistic structure for the parameters which is commonly available in many commonly used models.

We provide extensive simulation evidence to address the efficiency of PL versus standard methods. Computational time and accuracy are used to assess the performance. Our approach compares very favorably with these existing strategies and is robust to particle degeneracies as the sample size grows. Finally, PL has the additional advantage of being an intuitive and easy-to-implement computational scheme and should, therefore, become a default choice for posterior inference in a variety of models, with examples already appearing in Lopes et al. (2010), Carvalho et al. (2009), Prado and Lopes (2010), Lopes and Tsay (2010) and Lopes and Polson (2010).

ACKNOWLEDGMENTS

We thank the Editor, Raquel Prado, Peter Müller and Mike West for their invaluable comments that greatly improved the presentation of the ideas of the paper. R code for all examples are freely available upon request.

REFERENCES

BRIERS, M., DOUCET, A. and MASKELL, S. (2010). Smoothing algorithms for state-space models. *Ann. Inst. Statist. Math.* **62** 61–89. MR2577439

CAPPÉ, O., GODSILL, S. and MOULINES, E. (2007). An overview of existing methods and recent advances in sequential Monte Carlo. *IEEE Proceedings* **95** 899–924.

CARLIN, B., POLSON, N. G. and STOFFER, D. (1992). A Monte Carlo approach to nonnormal and nonlinear state-space modeling. *J. Amer. Statist. Assoc.* **87** 493–500.

CARTER, C. and KOHN, R. (1994). On Gibbs sampling for state space models. *Biometrika* **82** 339–350. MR1311096

CARVALHO, C. M. and LOPES, H. F. (2007). Simulation-based sequential analysis of Markov switching stochastic volatility models. *Comput. Statist. Data Anal.* **51** 4526–4542. MR2364463

CARVALHO, C. M., LOPES, H. F., POLSON, N. G. and TADDY, M. (2009). Particle learning for general mixtures. Working paper, Univ. Chicago Booth School of Business.

CHEN, R. and LIU, J. (2000). Mixture Kalman filters. *J. Roy. Statist. Soc. Ser. B* **62** 493–508. MR1772411

DOUCET, A., DE FREITAS, J. and GORDON, N. (2001). *Sequential Monte Carlo Methods in Practice*. Springer, New York. MR1847783

FEARNHEAD, P. (2002). Markov chain Monte Carlo, sufficient statistics, and particle filters. *J. Comput. Graph. Statist.* **11** 848–862. MR1951601

FEARNHEAD, P., WYNCOLL, D. and TAWN, J. (2008). A sequential smoothing algorithm with linear computational cost. Working paper, Dept. Mathematics and Statistics, Lancaster Univ.

FRÜHWIRTH-SCHNATTER, S. (1994). Applied state space modelling of non-Gaussian time series using integration-based Kalman filtering. *Statist. Comput.* **4** 259–269.

GAMERMAN, D. and LOPES, H. F. (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman & Hall/CRC Press, Boca Raton, FL. MR2260716

GODSILL, S. J., DOUCET, A. and WEST, M. (2004). Monte Carlo smoothing for nonlinear time series. *J. Amer. Statist. Assoc.* **99** 156–168. MR2054295

GORDON, N., SALMOND, D. and SMITH, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F* **140** 107–113.

JOHANNES, M. and POLSON, N. G. (2008). Exact particle filtering and learning. Working paper, Univ. Chicago Booth School of Business.

JOHANNES, M., POLSON, N. G. and YAE, S. M. (2008). Nonlinear filtering and learning. Working paper, Univ. Chicago Booth School of Business.

KALMAN, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering* **82** 35–45.

LIU, J. and CHEN, R. (1998). Sequential Monte Carlo methods for dynamic systems. *J. Amer. Statist. Assoc.* **93** 1032–1044. MR1649198

LIU, J. and WEST, M. (2001). Combined parameters and state estimation in simulation-based filtering. In *Sequential Monte Carlo Methods in Practice* (A. Doucet, N. de Freitas and N. Gordon, eds.). Springer, New York. MR1847793

LOPES, H. F. and POLSON, N. G. (2010). Extracting SP500 and NASDAQ volatility: The credit crisis of 2007–2008. In *Handbook of Applied Bayesian Analysis* (A. O'Hagan and M. West, eds.) 319–342. Oxford Univ. Press, Oxford.

LOPES, H. F. and TSAY, R. E. (2010). Bayesian analysis of financial time series via particle filters. *J. Forecast.* To appear.

LOPES, H. F., CARVALHO, C. M., JOHANNES, M. and POLSON, N. G. (2010). Particle learning for sequential Bayesian computation. In *Bayesian Statistics 9* (J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith and M. West, eds.). Oxford Univ. Press, Oxford.

PAPASPILIOPOULOS, O. and ROBERTS, G. (2008). Stability of the Gibbs sampler for Bayesian hierarchical models. *Ann. Statist.* **36** 95–117. MR2387965

PITT, M. and SHEPHARD, N. (1999). Filtering via simulation: Auxiliary particle filters. *J. Amer. Statist. Assoc.* **94** 590–599. MR1702328

POLSON, N. G., STROUD, J. and MÜLLER, P. (2008). Practical filtering with sequential parameter learning. *J. Roy. Statist. Soc. Ser. B* **70** 413–428. MR2424760

PRADO, R. and LOPES, H. F. (2010). Sequential parameter learning and filtering in structured autoregressive models. Working paper, Univ. Chicago Booth School of Business.

STORVIK, G. (2002). Particle filters in state space models with the presence of unknown static parameters. *IEEE Trans. Signal Process.* **50** 281–289.

WEST, M. (1986). Bayesian model monitoring. *J. Roy. Statist. Soc. Ser. B* **48** 70–78. MR0848052

WEST, M. and HARRISON, J. (1997). *Bayesian Forecasting and Dynamic Models*, 2nd ed. Springer, New York. MR1482232